

CSE211 Dijital Tasarım

Akdeniz Üniversitesi

Hafta 7-8-9: Kapı Seviyesi En Aza İndirme

1

Doç.Dr. Taner Danışman
tdanisman@akdeniz.edu.tr

2

Ders programı

Hafta 01	09/16/2024 Giriş
Hafta 02	23/09/2024 Dijital Sistemler ve İkili Sayılar I
Hafta 03	30/09/2024 Dijital Sistemler ve İkili Sayılar II
Hafta 04	10/07/2024 Boole Cebiri ve Mantık Kapıları I
Hafta 05	10/14/2024 Boole Cebiri ve Mantık Kapıları II
Hafta 06	10/21/2024 Kapı Seviyesi Minimizasyonu
Hafta 07	10/28/2024 Karnaugh Haritaları
Hafta 08	11/04/2024 Karnaugh Haritaları
Hafta 09	11/11/2024 Vize
Hafta 10	11/18/2024 Kombinasyonel Mantık
11. Hafta	25.11.2024 Kombinasyonel Mantık
12. Hafta	12/02/2024 Zamanlama, gecikmeler ve tehlikeler
Hafta 13	12/09/2024 Eşzamanlı Sıralı Mantık
Hafta 14	12/16/2024 Eşzamanlı Sıralı Mantık

3

Bilmeniz Gereken Bölüm Terimleri

Karnaugh Haritası

Kablolu Mantık

Koşulları önemsemeyin

HDL

4

Kapı Seviyesi Minimizasyonu

Kapı düzeyinde en aza indirme, bir sayısal devreyi tanımlayan Boole fonksiyonlarının kapı düzeyinde optimum uygulamasını bulma tasarım görevidir.

Boole fonksiyonlarının basitleştirilmesi daha basit (ve genellikle daha hızlı) dijital devrelere yol açar.

Sorun:

Problemin birden fazla bileşeni olduğunda mantık tasarımında zorluk girdiler

Birbirini izleyen adımları tahmin etmek için belirli kurallardan yoksundur basitleştirme süreci

Kimlikleri kullanarak Boole işlevlerini basitleştirmek zaman alıcıdır ve hataya açık.

Neden öğrenme?

Bir tasarımcının sorunun altında yatan matematiksel açıklamayı ve çözümü anlaması önemlidir

5

Harita Yöntemi

Benzersiz Doğruluk tabloları cebirsel olarak temsil edildiğinde, birçok farklı, ancak eşdeğer formda görünebilir.

Minimizasyon zor bir süreçtir.

Harita yöntemi :

Burada sunulan **Map** yöntemi, Boole fonksiyonlarını en aza indirmek için basit ve anlaşılır bir prosedür sağlar.

Ayrıca bir doğruluk tablosunun resimsel biçimi olarak da adlandırılır

Karnaugh **haritası** veya **K-haritası** olarak da bilinir .

6

K-haritası

K-haritası, her karenin en aza indirilecek fonksiyonun bir mintermini temsil ettiği karelерden oluşan bir diyagramdır .

Harita, bir kişinin tüm olası yollarının görsel bir diyagramını sunar . fonksiyon standart formda ifade edilebilir

Kullanıcı çeşitli desenleri tanıyarak,
aynı fonksiyon için alternatif cebirsel ifadeler , bunlardan en basit olanı
seçilebilir

K-haritası

Haritanın ürettiği basitleştirilmiş ifadeler her zaman aşağıdakilerden birindedir: iki standart form:

[ürünlerin toplamı](#) veya [toplamların çarpımı](#)

En basit cebirsel ifadenin şu şekilde olduğu varsayılacaktır:

[minimum terim sayısına](#) sahip ve
Her terimdeki mümkün olan [en küçük harf sayısını](#).

Bu ifade, [minimum sayıda kapı](#) ve her kapıya [minimum sayıda giriş](#) içeren bir devre şeması üretir

Bazen en basit ifade bile tek değildir:

iki veya daha fazla ifade en aza indirme kriterini karşılayabilir.

Bu gibi durumlarda her iki çözüm de yeterlidir.

Kmaps'in Tanımı ve Terminolojisi

Örneğin, aşağıdaki fonksiyona ait mintermler:

x ve y girdileri şunlardır: $F(x, y) = xy + x\bar{y}$

Boole fonksiyonunu ele alalım,

Mintermleri şunlardır:

$\bar{x}\bar{y}$, $\bar{x}y$, $x\bar{y}$, and xy

Minterm	X	Y
$\bar{x}\bar{y}$	0	0
$\bar{x}y$	0	1
$x\bar{y}$	1	0
xy	1	1

Kmaps'in Tanımı ve Terminolojisi

Bir Kmap'in her minterm için bir hücresi vardır.

Bu, bir fonksiyonun doğruluk tablosunun her satırı için bir hücreye sahip olduğu anlamına gelir.

$F(x,y) = xy$ fonksiyonuna ait doğruluk tablosu, karşılık gelen Kmap'ıyla birlikte sağda gösterilmektedir.

$$F(X, Y) = XY$$

X	Y	XY
0	0	0
0	1	0
1	0	0
1	1	1

		Y	0	1
		X	0	1
0	0	0	0	0
	1	0	0	1

10

K-Harita Kuralları

Birleştirilecek bitişik karelerin sayısı her zaman 1, 2, 4 ve 8 gibi iki kuvveti olan bir sayıyı temsil etmelidir.

Daha fazla sayıda bitişik kare birleştirilirse, daha az sayıda sabit içeren bir ürün terimi elde ederiz

1 kare = 1 minterm = üç sabit.

2 bitişik kare = 1 terim = iki sabit.

4 bitişik kare = 1 terim = bir sabit.

8 bitişik kare tüm haritayı çevreler ve her zaman 1'e eşit olan bir fonksiyon üretir .

Yani 1,2,4,8 gibi iki'nin kuvvetleri şeklinde birleştirilen bitişik karelerin sayısının bilinmesi gereği açıktır .

11

K-Harita Kuralları

Grup içinde sıfır yok

		0	1
		0	
A	B	0	
		1	

WRONG

		0	1
		0	
A	B	0	
		1	

RIGHT

Gruplar dikey veya yatay olabilir ancak çapraz olamaz

		0	1
		0	
A	B	0	
		1	

WRONG

		0	1
		0	
A	B	0	
		1	

RIGHT

12

K-Harita Kuralları (Devamı)

Gruplar örtüşmeli

		AB	00	01	11	10	
		C	0	(1)	1	1	1
C	0	0	0	0	1	1	
		1	0	0	1	1	

Groups overlapping.

RIGHT ✓

		AB	00	01	11	10	
		C	0	(1)	(1)	(1)	(1)
C	0	0	0	0	1	1	
		1	0	0	1	1	

Groups not overlapping.

WRONG ✗

13

K-Harita Kuralları (Devamı)

Grupların 2^n 1'i olmalıdır . ($n \geq 0$)

Group of 2

A	B	0	1
0	1	1	1
1	0	0	0

RIGHT ✓

Group of 3

A	B	C	00	01	11	10
0	0	0	0	1	1	1
1	0	0	0	0	0	0

WRONG ✗

Group of 4

A	B	0	1
0	1	1	1
1	1	1	1

RIGHT ✓

Group of 5

A	B	C	00	01	11	10
0	1	1	1	1	1	1
1	0	0	0	0	0	1

WRONG ✗

14

K-Harita Kuralları (Devamı)

Gruplar, 2 kişi düşünülerek mümkün olduğunca büyütülmelidir .
kural

A B
C
00 01 11 10

1	1	1	1
0	0	1	1
0	1	1	1

RIGHT ✓

A B
C
00 01 11 10

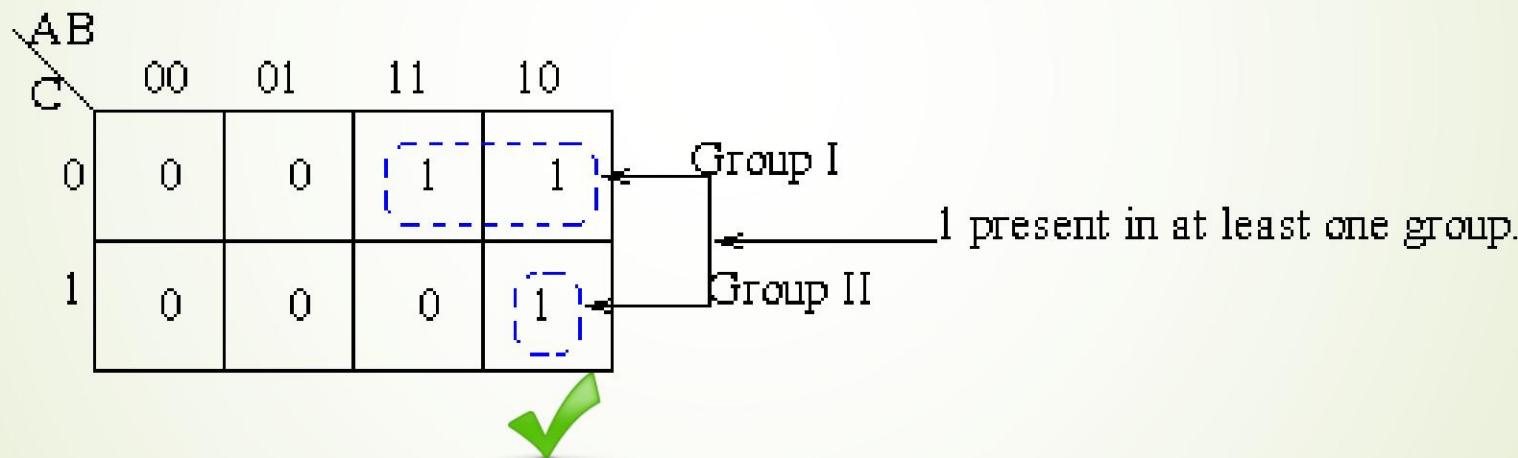
1	1	1	1
0	0	1	1
1	1	1	1

WRONG ✗

15

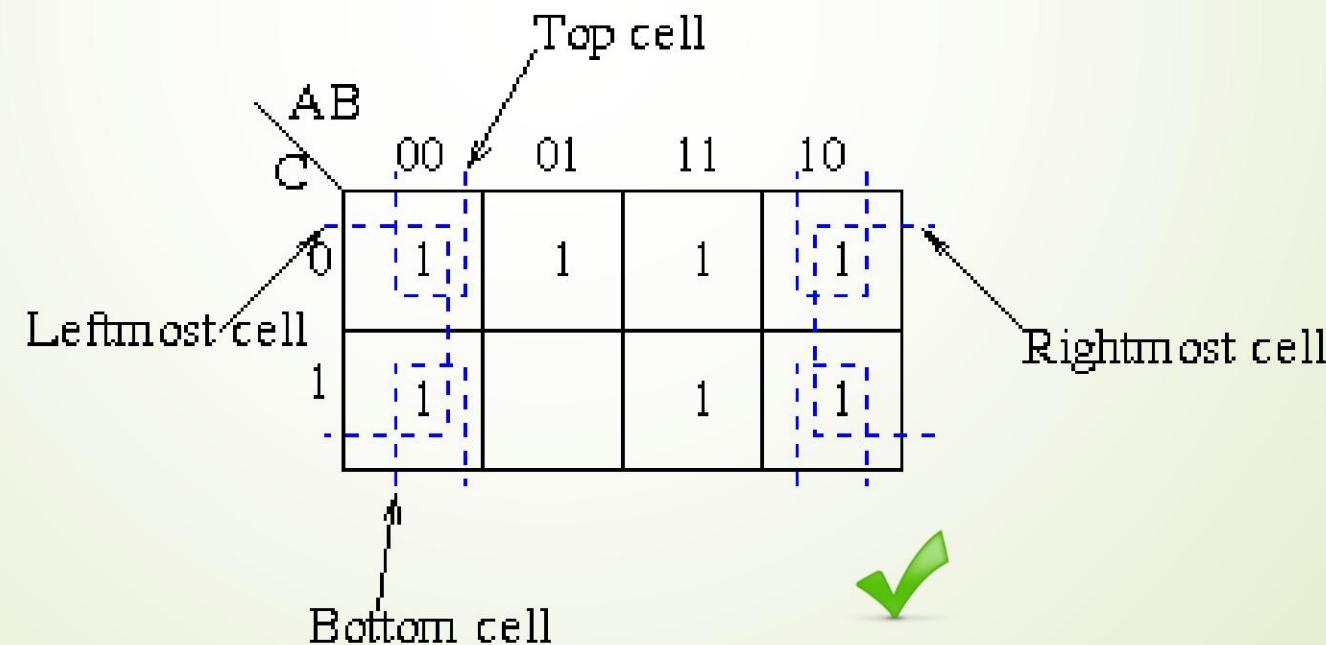
K-Harita Kuralları (Devamı)

Her 1 en az bir grupta sunum yapmalıdır



K-Harita Kuralları (Devamı)

Gruplar üstten, alttan,
Bir bitlik Gray kodunu tatmin eden en sağdaki ve en soldaki
 hücreler değişir.



İki Değişken İçin Kmap Basitleştirmesi

Kmap sadeleştirmesinin kuralları şunlardır:

- Gruplamalar yalnızca 1'lerden oluşabilir; 0'lardan oluşamaz.
- Gruplar sadece dik açılarla oluşturulabilir; çapraz gruplar oluşturulamaz.
- Bir gruptaki 1'lerin sayısı 2'nin bir kuvveti olmalıdır – tek bir 1 bile içerse.
- Gruplar mümkün olduğunca büyük oluşturulmalıdır.
- Gruplar üst üste gelebilir ve yanlara doğru sarılabilir.

K-Harita.

İki Değişkenli Harita

İki değişkenli dört minterm vardır ve dörtten oluşur kareler.

$$m_1 + m_2 + m_3 = x'y + xy' + xy = x + y$$

m_0	m_1
m_2	m_3

(a)

		y	
		0	1
x	0	m_0 $x'y'$	m_1 $x'y$
	1	m_2 xy'	m_3 xy

(b)

		y	
		0	1
x	0		
	1		1

(a) xy

		y	
		0	1
x	0		1
	1	1	1

(b) $x + y$

Fig. 3-2 Representation of Functions in the Map

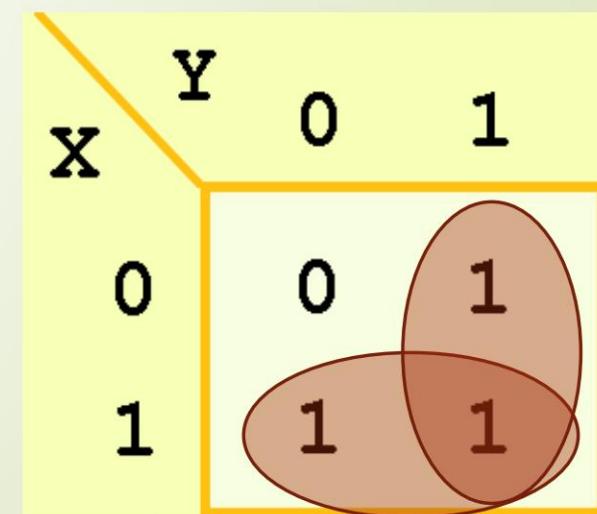
19

İki Değişken İçin Kmap Basitleştirmesi

Elbette Kmap'imizden türettiğimiz minterm fonksiyonu en basit haliyle değildi. Bu örnekte başladığımız şey buydu.

Ancak, Kmap'te bitişik 1'leri bularak karmaşık ifademizi en basit terimlere indirgeyebiliriz
İki'nin kuvvetleri olan gruplara toplanabilen .

- Örneğimizde iki tane böyle grubumuz var.



20

İki Değişkenli Harita

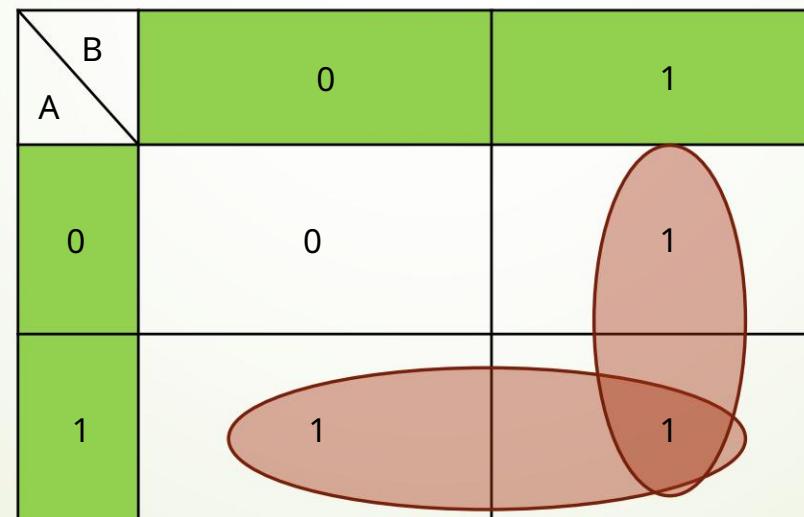
VEYA Örneği

Gray kodu kullanılır

$$F(A,B)=A'B+AB'+AB=A+B$$

Gerçeklik Tablosu
 $S=A+B$

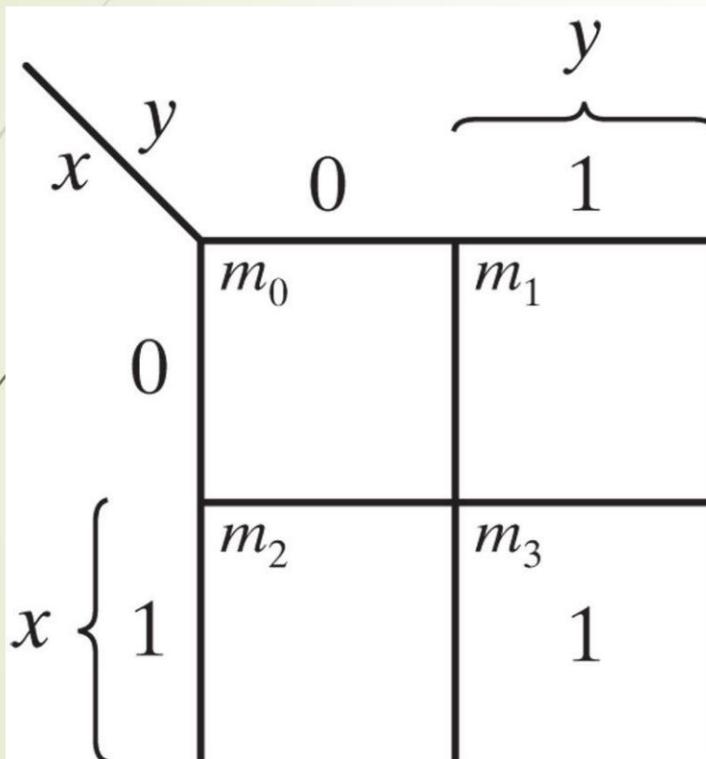
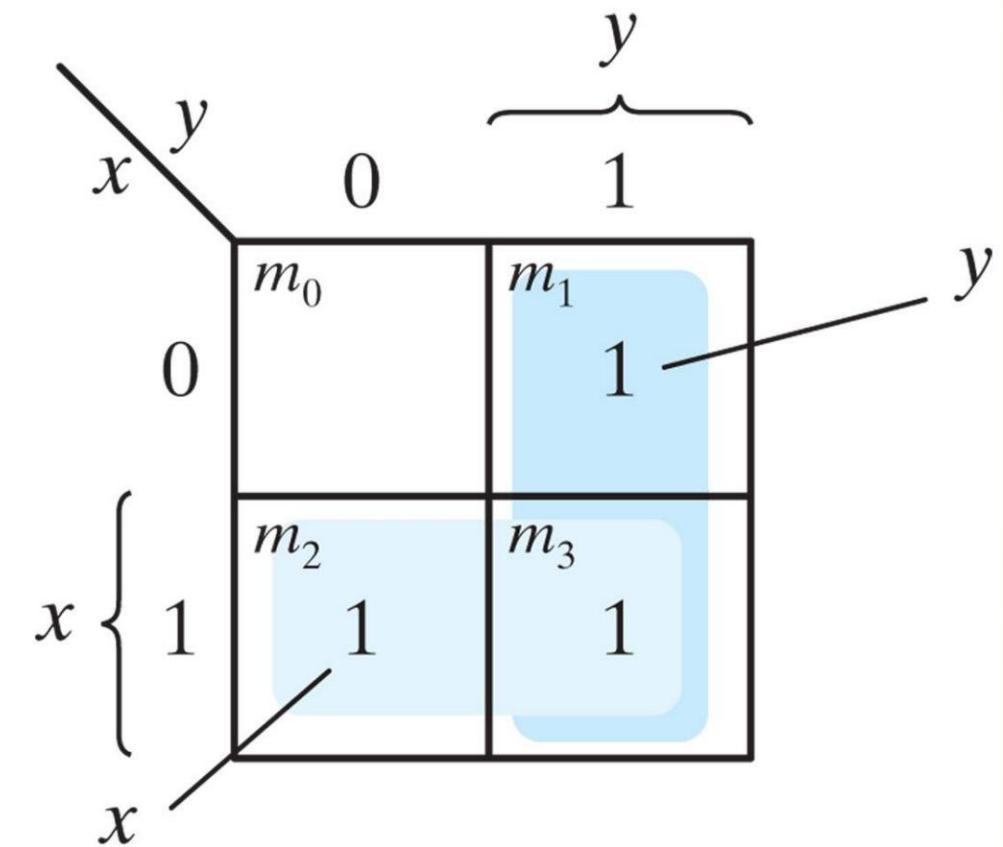
A	B	S
0	0	0
0	1	1
1	0	1
1	1	1



21

Fonksiyonların Gösterimleri

Harita

(a) xy (b) $x + y$

Üç Değişkenli Harita

Mintermlerin ikili bir dizide değil , Gray koduna benzer şekilde düzenlendiğine dikkat edin .

sadece bir bit, bitişiktekinden değer olarak değişir
sütundan bir sonrakine

Örn: m5'e atanın kare, satır 1 ve sütun 01'e karşılık gelir

m_0	m_1	m_3	m_2
m_4	m_5	m_7	m_6

(a)

		00	01	y		
		x ^{yz}	x ^{y'z'}	11	10	
		0	m_0	m_1	m_3	m_2
x	1	m_4	m_5	m_7	m_6	
		$xy'z'$	$xy'z$	xyz	xyz'	

(b)

23

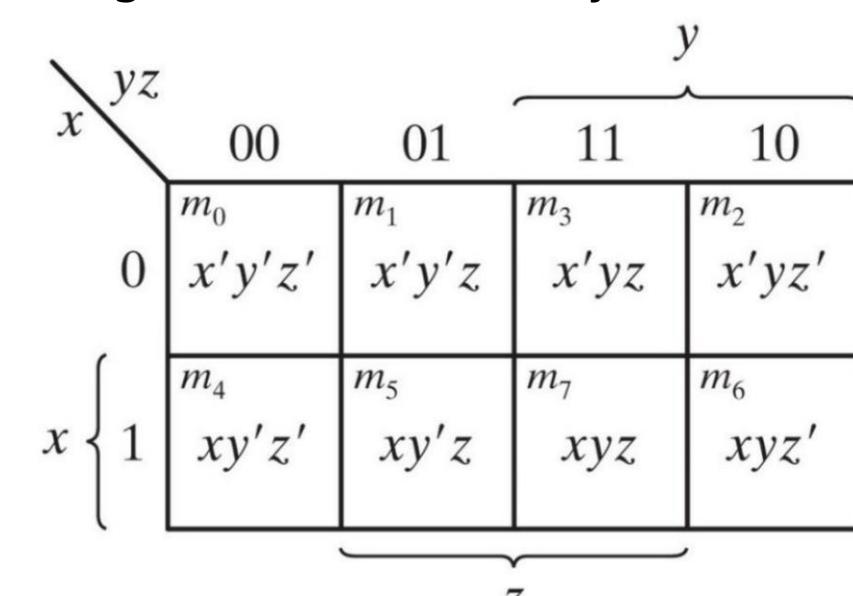
Üç Değişkenli Harita

Bu nedenle, K-haritasının ilk satırı tüm mintermleri içerir x 'in değeri sıfırdır .

İlk sütun, y ve z 'nin her ikisinin de olduğu tüm mintermleri içerir sıfır değerine sahiptir.

m_0	m_1	m_3	m_2
m_4	m_5	m_7	m_6

(a)



(b)

24

Üç Değişkenli Harita

Boole fonksiyonlarını basitleştirmek için, bitişik karelerin sahip olduğu temel özelliği tanımlıyoruz .

Haritadaki herhangi iki bitişik kare yalnızca bir farkla farklıdır değişken

Bitişik karelerdeki iki mintermin toplamı şu şekilde olabilir : sadece iki sabit değerden oluşan tek bir ürün terimine basitleştirilmiştir .

ÖRNEĞİN:

$$m_5 + m_7 = xy'z + xyz = xz(y' + y) = xz$$

Bu nedenle, bitişik karelerdeki (dikey veya yatay olarak, ancak çapraz olarak değil, bitişik) OR'lu birlikte farklı değişkenin kaldırılmasına neden olacaktır

25

Üç Değişken İçin Kmap Basitleştirmesi

Şimdi daha karmaşık bir Kmap için. Şunu düşünün işlev:

$$F(X, Y, Z) = \bar{X}\bar{Y}\bar{Z} + \bar{X}\bar{Y}Z + \bar{X}YZ + \bar{X}Y\bar{Z} + X\bar{Y}\bar{Z} + XY\bar{Z}$$

Kmap'i aşağıda gösterilmiştir. (Sadece) iki tane vardır 1'lerin gruplandırılması.

Onları bulabilir misin?

		YZ	00	01	11	10
		X	0	1	1	1
X	0	1	1	1	1	1
	1	1	0	0	1	

26

Üç Değişken İçin Kmap Basitleştirmesi

Bu Kmap'te, bir Kmap'in kenarlarını saran bir grubun örneğini görüyoruz.

Bu grup bize x ve y değerlerinin, grubun kapsadığı fonksiyonun terimiyle ilgili olmadığını söyler.

Bu bize fonksiyonun bu terimi hakkında ne anlatıyor?

Peki ya en üst sıradaki
yeşil grup?

x	y	z	00	01	11	10
0	0	0	1	1	1	1
1	1	0	1	0	0	1

27

Üç Değişken İçin Kmap Basitleştirmesi

En üst sıradaki yeşil grup bize yalnızca

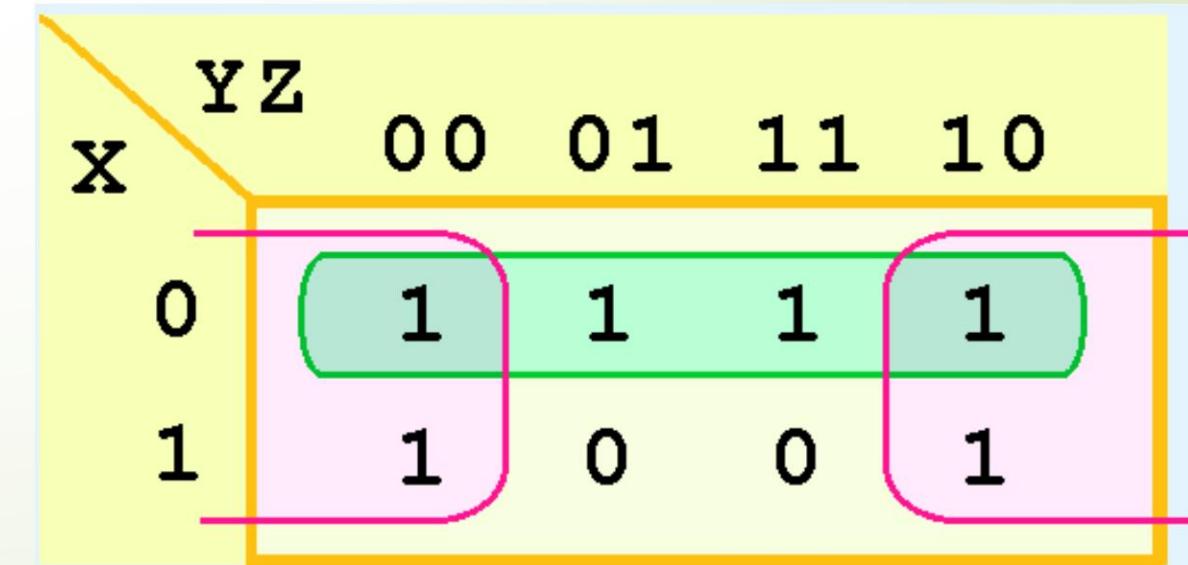
x değeri 0 grupta anlamlıdır.

O satırda tamamlayıcı olduğunu görüyoruz, yani diğer indirgenmiş fonksiyonun terimi \bar{x} .

İndirgenmiş fonksiyonumuz şudur:

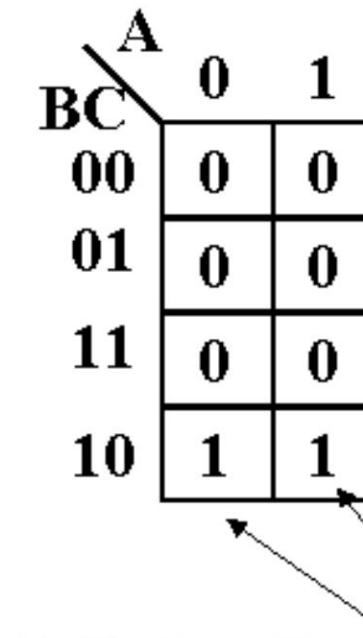
$$F(X, Y, Z) = \bar{X} + \bar{Z}$$

Altı tane mintermimiz olduğunu hatırlayın
orijinal fonksiyonumuzda !



Üç Değişkenli Harita

Row	A B C	F(A,B,C)
0	0 0 0	0
1	0 0 1	0
2	0 1 0	1
3	0 1 1	0
4	1 0 0	0
5	1 0 1	0
6	1 1 0	1
7	1 1 1	0



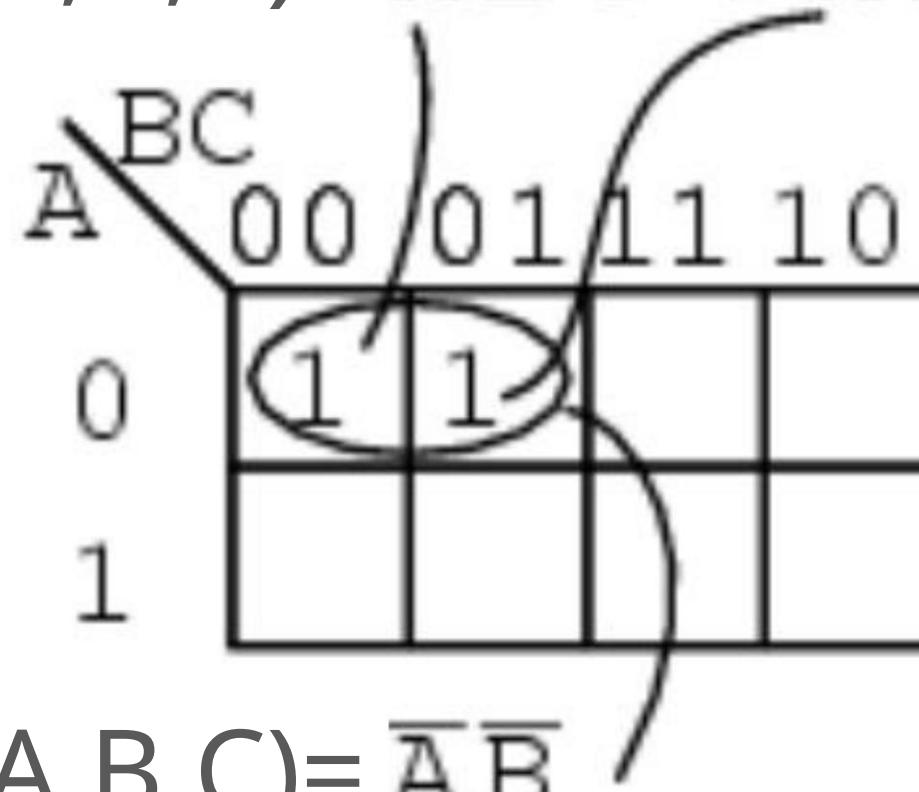
$$\begin{aligned}
 F(A, B, C) &= \sum m(2, 6) \\
 &= A'BC' + ABC' \\
 &= BC'(A' + A) \\
 &= BC'
 \end{aligned}$$

Boolean adjacency can be used to minimize functions!

29

Üç değişkenli K-Harita Örneği #1

$$F(A,B,C) = \overline{A}\overline{B}\overline{C} + \overline{A}\overline{B}C$$



30

Örnek 3.1 Üç Değişkenli Harita

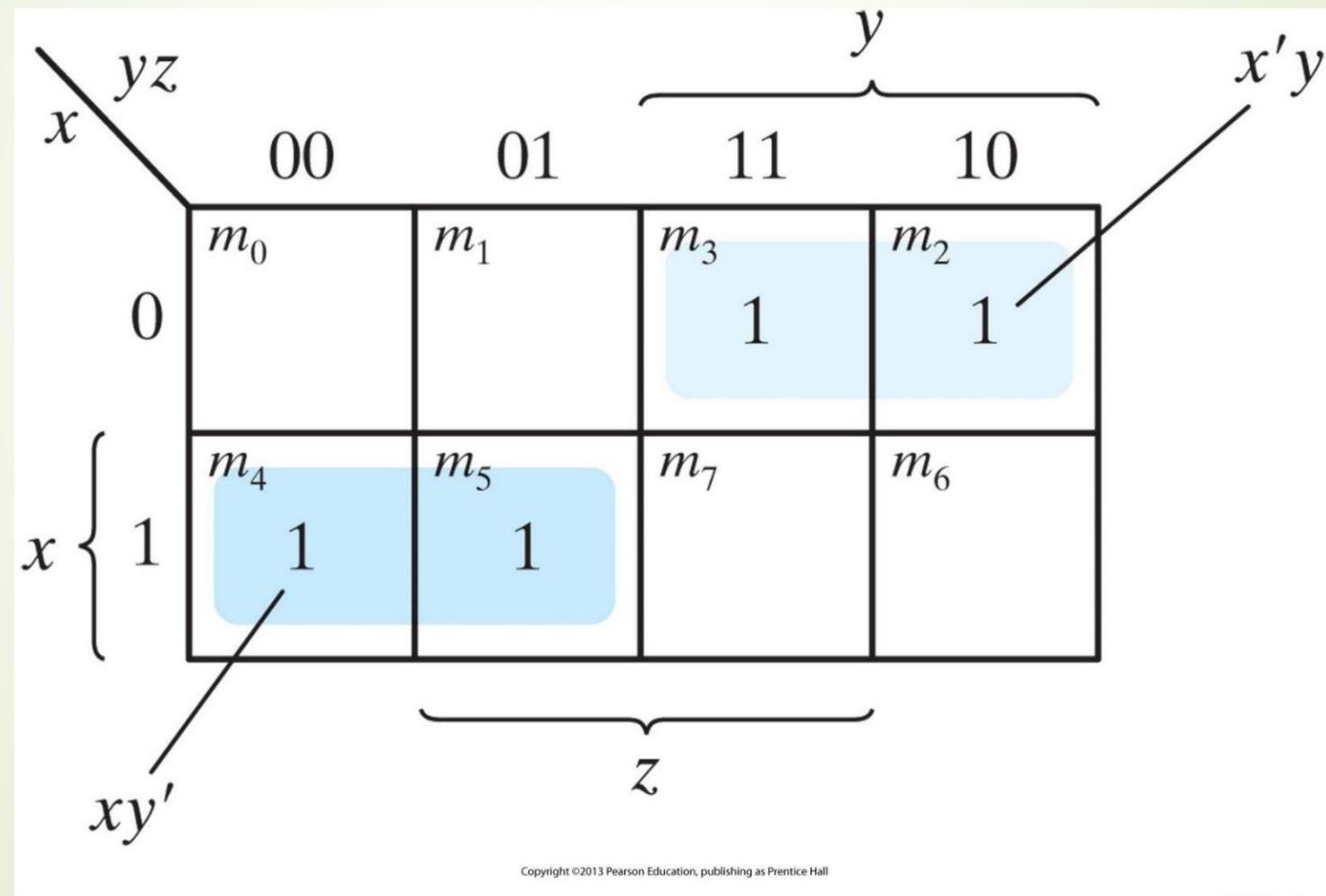
EXAMPLE 3.1

Simplify the Boolean function

$$F(x, y, z) = \Sigma(2, 3, 4, 5)$$

31

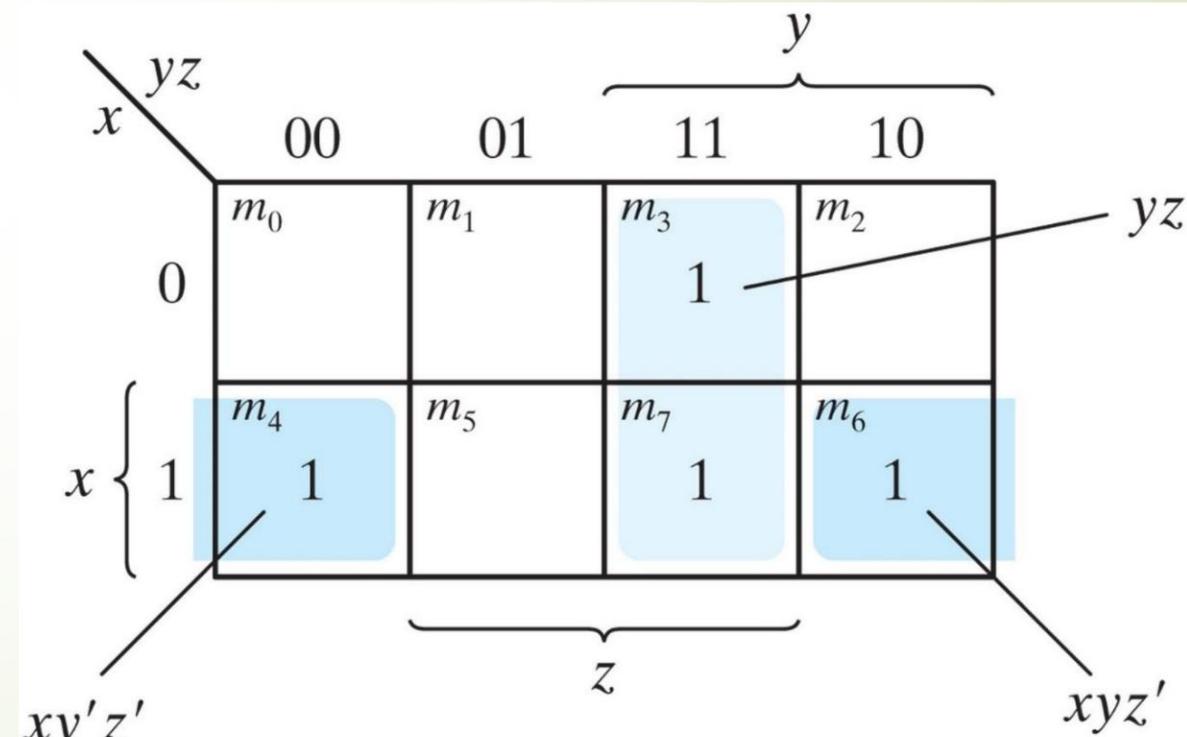
Örnek 3.1 için Harita, $F(x, y, z) = \Sigma(2, 3, 4, 5) = x'y + xy$



32

Örnek 3.2 $F(x, y, z) = \sum(3, 4, 6, 7) = yz + xz'$

Bazı durumlarda, iki
Haritadaki kareler bitişik olarak
kabul edilir, ancak
birbirlerine dokunmuyorlar



33

Üç değişkenli haritada dört bitişik karenin birleşimi

Bu tür birleşimler dört mintermin mantıksal toplamını temsil eder ve **yalnızca bir sabit** içeren bir ifadeyle sonuçlanır

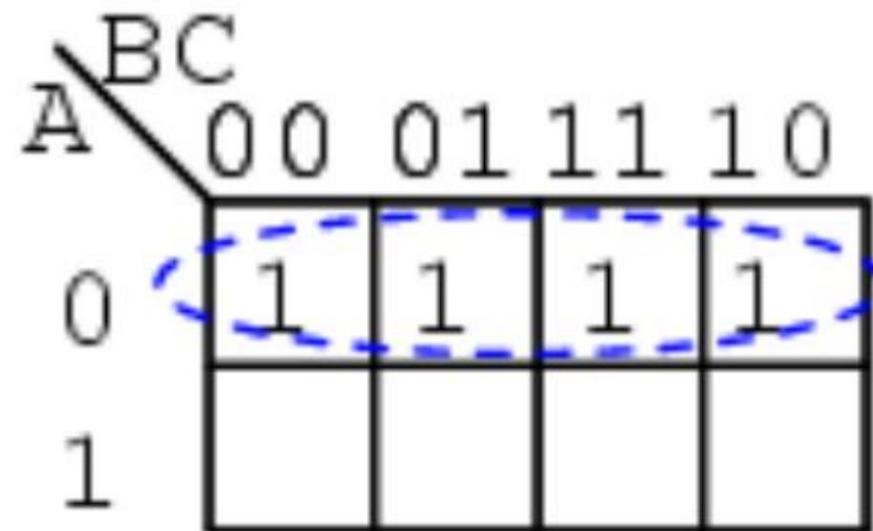
Örnek olarak, dört bitişik sayının mantıksal toplamı mintermler

$$\begin{aligned}0, 2, 4 \text{ ve } 6, \text{ tek bir literal terim olan } z' \text{ye indirgenir: } & m_0 + \\m_2 + m_4 + m_6 = x'y'z' + x'yz' + xy'z' + xyz' \\&= x'z'(y' + y) + xz'(y' + y) = x'z' \\&+ xz' = z'(x' + x) = z'\end{aligned}$$

34

Üç değişkenli K-Harita Örneği #2

$$F(A,B,C) = \overline{A}\overline{B}\overline{C} + \overline{A}\overline{B}C + \overline{A}BC + \overline{A}B\overline{C}$$



$$F(A,B,C) = \overline{A}$$

35

Üç değişkenli K-haritası Örneği #3

$$F(A,B,C) = \overline{A}\overline{B}C + \overline{A}BC + A\overline{B}C + ABC$$

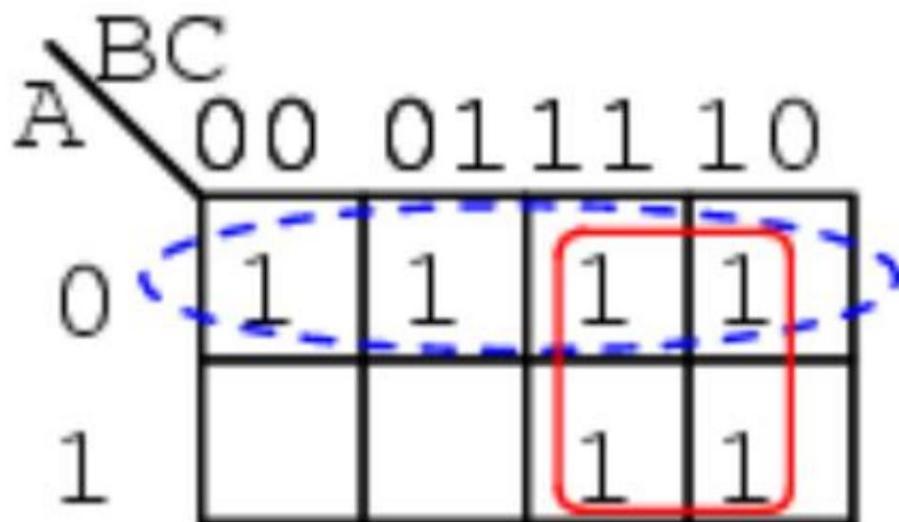
		BC	A	00	01	11	10
		A	0				
		B	0				
		0			1	1	
		1			1	1	

$$F(A,B,C) = C$$

36

Üç değişkenli K-haritası Örneği #4

$$F(A,B,C) = \overline{A}\overline{B}\overline{C} + \overline{A}\overline{B}C + \overline{A}BC + \overline{A}B\overline{C} + ABC + A\overline{B}\overline{C}$$



$$F(A,B,C) = \overline{A} + B$$

37

K-haritası Örneği #5 üç değişkenli

$$F(A,B,C) = \overline{A}BC + \overline{A}B\overline{C} + ABC + AB\overline{C}$$

A	BC	00	01	11	10
0				1	1
1				1	1

$$F(A,B,C) = B$$

38

Üç değişkenli K-haritası Örneği #6

A M.Ö.	00	01	11	10
0		1	1	
1		1	1	1

$$A(B, C, C) = C + AB$$

39

Örnek 3.3

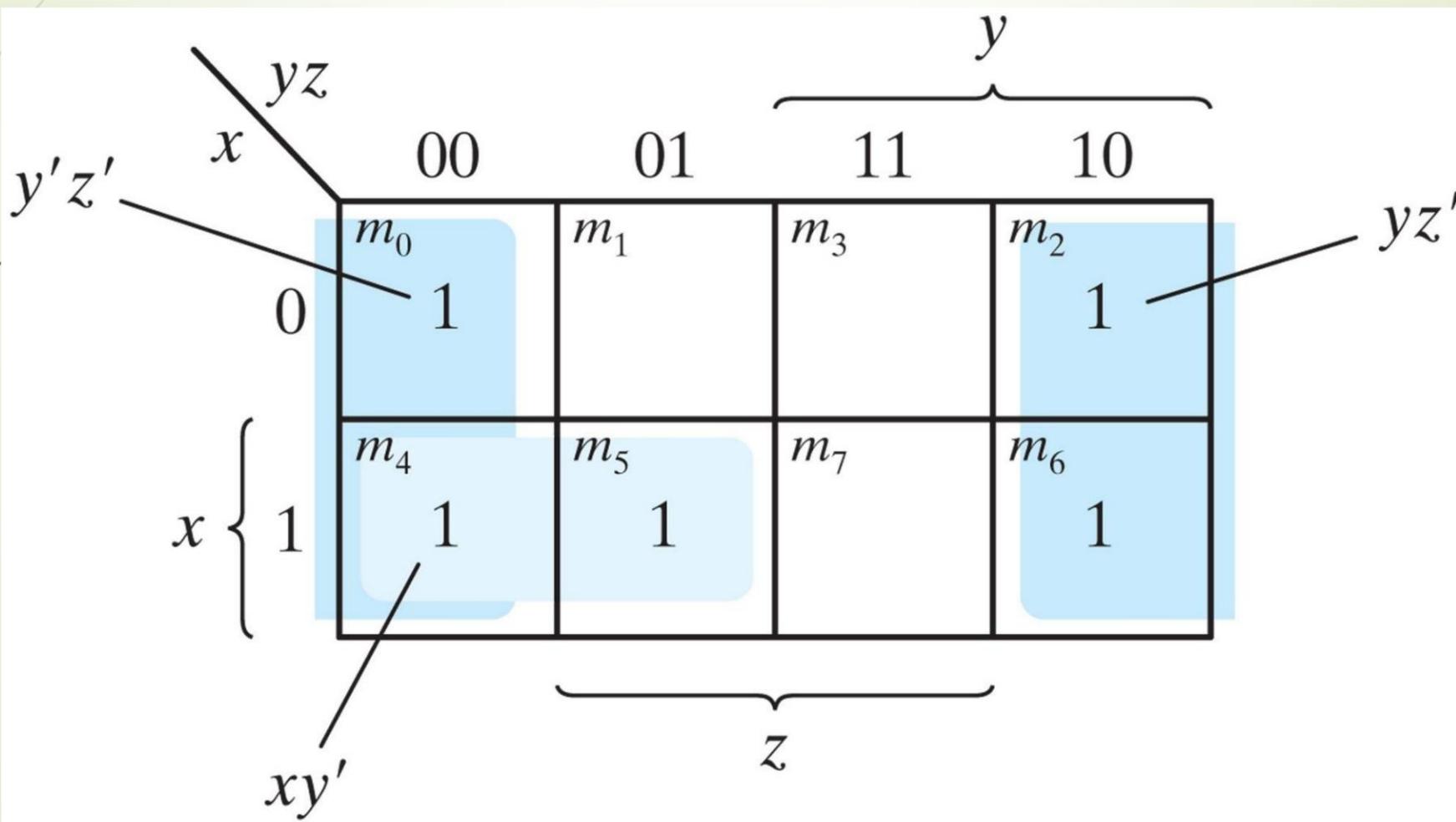
EXAMPLE 3.3

Simplify the Boolean function

$$F(x, y, z) = \Sigma(0, 2, 4, 5, 6)$$

40

Örnek 3.3 için Harita, $F(x, y, z) = \Sigma(0, 2, 4, 5, 6) = z' + xy'$



41

Eksik giriş sabitleri

Bir fonksiyon, mintermlerin toplamı biçiminde ifade edilmezse, fonksiyonun mintermlerini elde etmek için haritayı kullanmak ve ardından fonksiyonu , en az sayıda terime sahip bir ifadeye basitleştirmek mümkündür.

Örnek 3.4

EXAMPLE 3.4

For the Boolean function

$$F = A'C + A'B + AB'C + BC$$

- Express this function as a sum of minterms.
- Find the minimal sum-of-products expression.

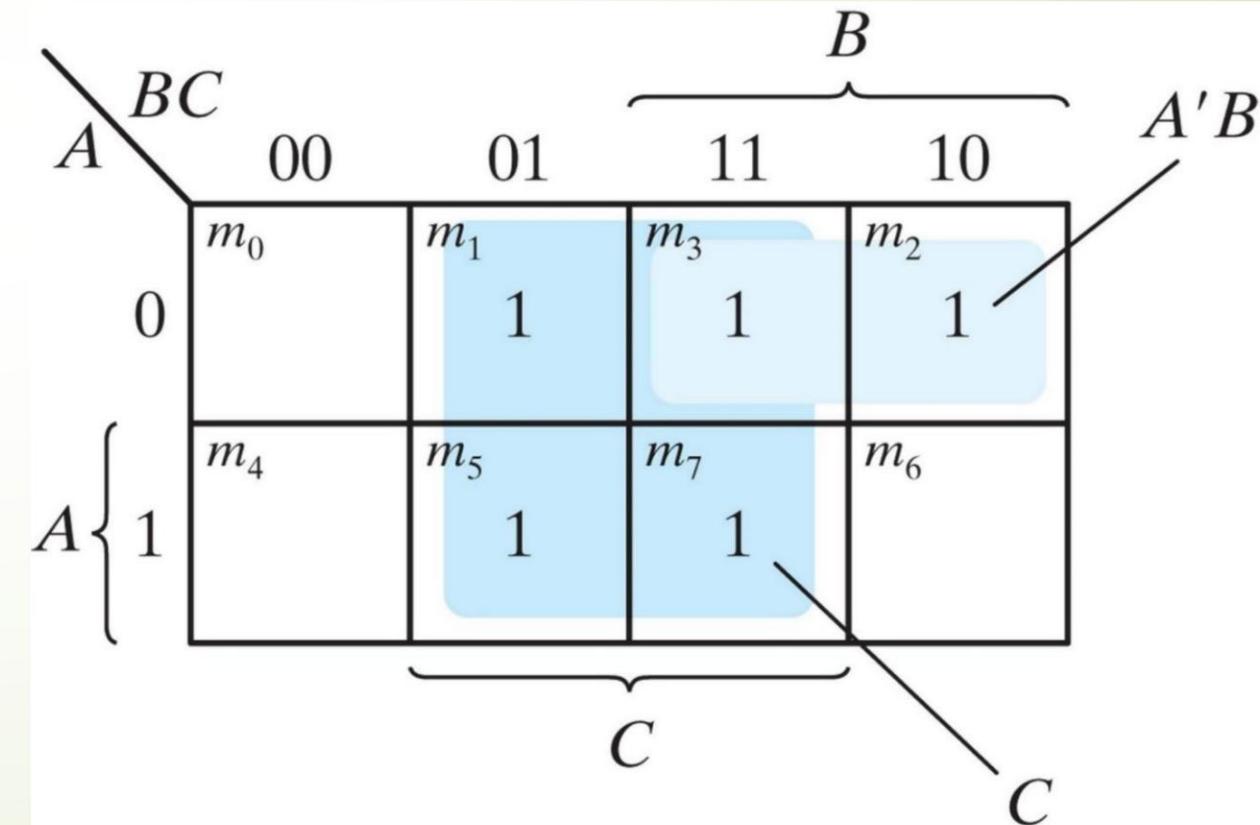
43

Örnek 3.4'ün Haritası, $A'C + A'B + AB'C + BC = C + A'B$

$$F(A, B, C) = (1, 2, 3, 5, 7)$$

Başlangıçta verilen
ürün toplamı ifadesi çok
fazla
Şartlar

$$F = C + A'B$$



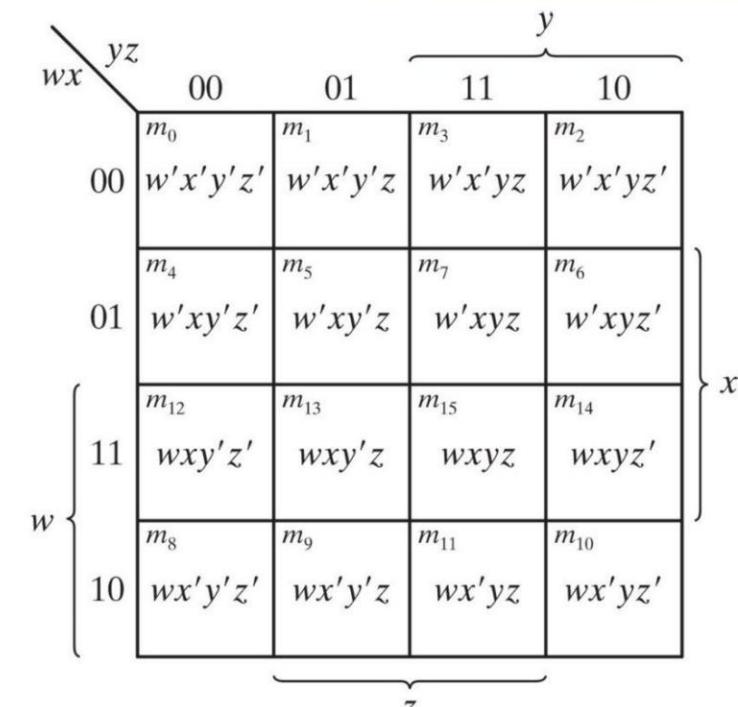
Dört Değişkenli K-haritası

Üçüncü satırın (11) ve ikinci sütunun (01) sayıları birleştirildiğinde, ondalık 13'ün ikili eşdeğeri olan 1101 ikili sayısını verir.

Bu nedenle, üçüncü satır ve ikinci sütundaki kare minterm m_{13} 'ü temsil eder

m_0	m_1	m_3	m_2
m_4	m_5	m_7	m_6
m_{12}	m_{13}	m_{15}	m_{14}
m_8	m_9	m_{11}	m_{10}

(a)



(b)

Dört Değişkenli K-haritası

1 kare = 1 minterm = 4 değişmez

2 bitişik kare = 1 terim = 3 değişmez 4

bitişik kare = 1 terim = 2 değişmez 8 bitişik

kare = 1 terim = 1 değişmez 16 bitişik

kare = 1

46

Örnek 3.5 Dört Değişkenli K-haritası

EXAMPLE 3.5

Simplify the Boolean function

$$F(w, x, y, z) = \Sigma(0, 1, 2, 4, 5, 6, 8, 9, 12, 13, 14)$$

47

Örnek 3.5

$$F(w, x, y, z)$$

$$= \Sigma(0,1,2,4,5,6,8,9,12,13,14)$$

$$= y' + w'z' + xz'$$

A Karnaugh map for four variables w, x, y, z . The columns are labeled wx and yz , and the rows are labeled $w'y'z'$, $w'yz'$, $xy'z'$, and xyz' . The map shows minterms $m_0, m_1, m_2, m_3, m_4, m_5, m_6, m_7, m_8, m_9, m_{10}, m_{11}, m_{12}, m_{13}, m_{14}, m_{15}$. Shaded regions represent the terms in the minimized expression $y' + w'z' + xz'$:

- $w'yz'$ (minterms m_0, m_1, m_2)
- $w'y'z'$ (minterms m_4, m_5, m_6)
- $xy'z'$ (minterms m_{12}, m_{13}, m_{15})
- xyz' (minterm m_{14})

Note: $w'yz' + w'yz' = w'z'$
 $xy'z' + xyz' = xz'$

48

Örnek 3.6

EXAMPLE 3.6

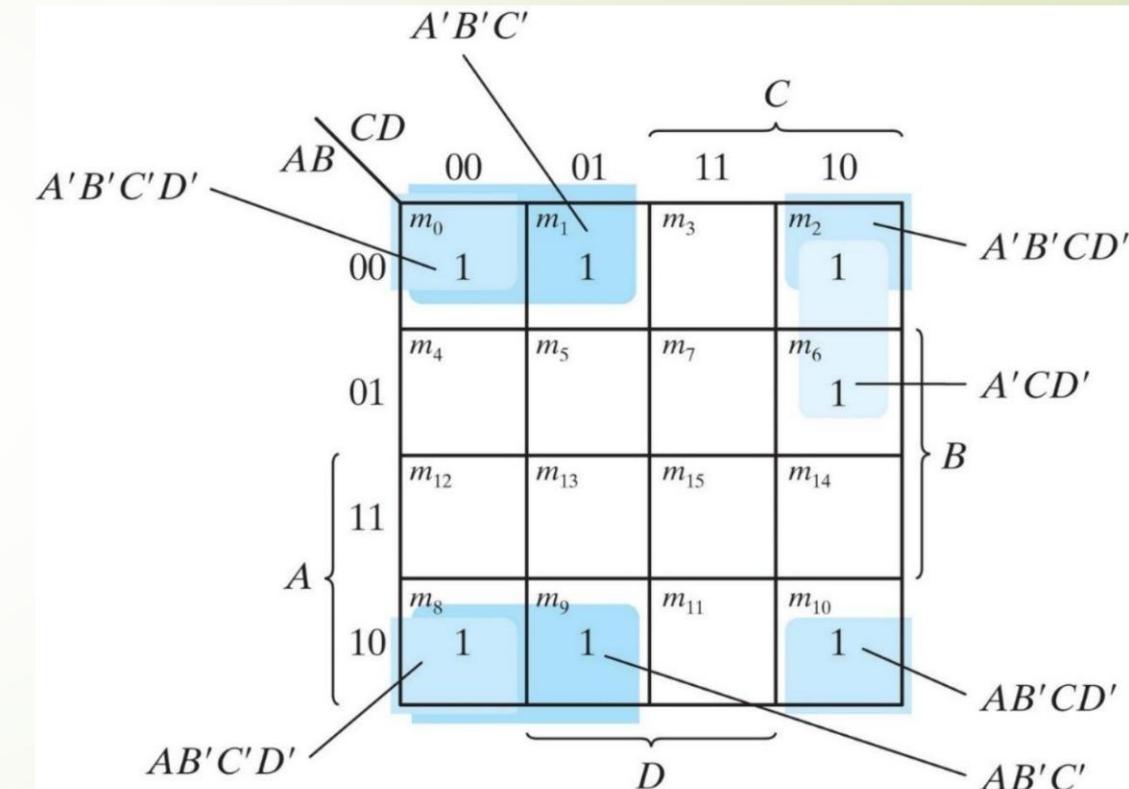
Simplify the Boolean function

$$F = A'B'C' + B'CD' + A'BCD' + AB'C'$$

49

Örnek 3.6

$$\begin{aligned}
 & A'B'C' + B'CD' + A'BCD' + AB'C' \\
 & = B'D' + B'C' + A'CD'
 \end{aligned}$$



Note: $A'B'C'D' + A'CD' = A'B'D'$
 $A'B'C'D' + AB'CD' = AB'D'$
 $A'B'D' + AB'D' = B'D'$
 $A'B'C' + AB'C' = B'C'$

Dört Değişken için K-haritası Basitleştirmesi

50

Aşağıda gösterilen Kmap'i aşağıdakilerle doldurduk:
fonksiyondan sıfır olmayan mintermler:

$$\begin{aligned} F(W, X, Y, Z) = & \bar{W}\bar{X}\bar{Y}\bar{Z} + \bar{W}\bar{X}\bar{Y}Z + \bar{W}\bar{X}YZ \\ & + \bar{W}XY\bar{Z} + W\bar{X}\bar{Y}\bar{Z} + W\bar{X}\bar{Y}Z + W\bar{X}YZ \end{aligned}$$

Bu Kmap'te (sadece) üç grubu tanımlayabilir misiniz?

Hatırlayın ki
gruplar
örtüsebilir.

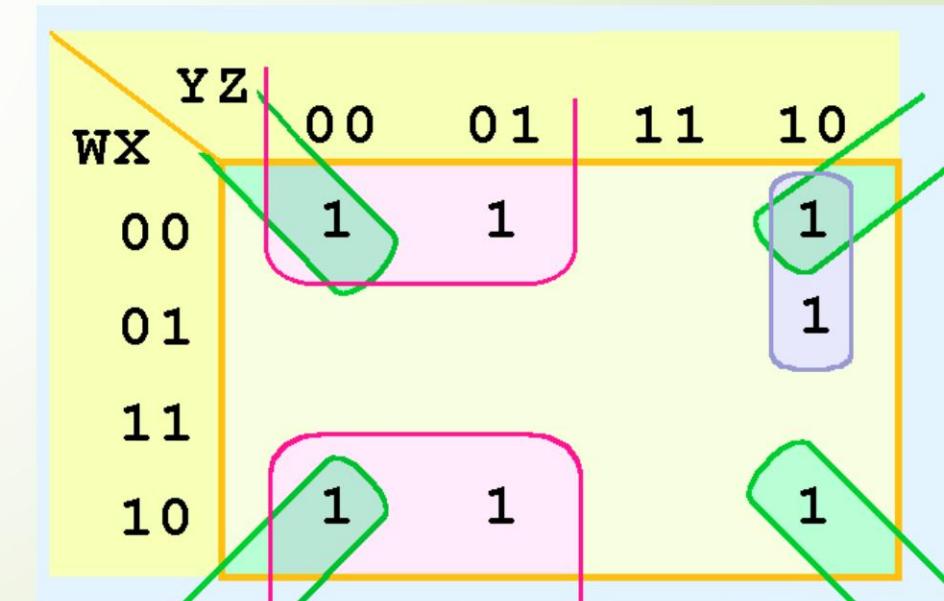
WX	YZ	00	01	11	10
00		1	1		1
01					1
11					
10		1	1		1

Dört Değişken İçin Kmap Basitleştirmesi

51

Üç grubumuz şunlardan oluşur: Sağdaki Kmap'in tamamen içinde olan mor bir grup. Üst ve alt kısımları saran pembe bir grup. Köşeleri kaplayan yeşil bir grup. Bu nedenle son fonksiyonumuzda üç terimimiz var:

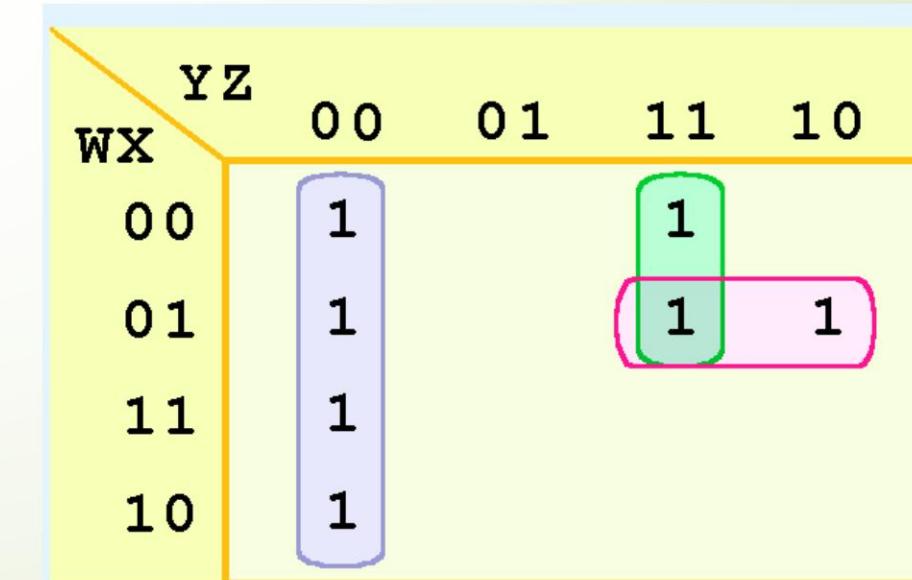
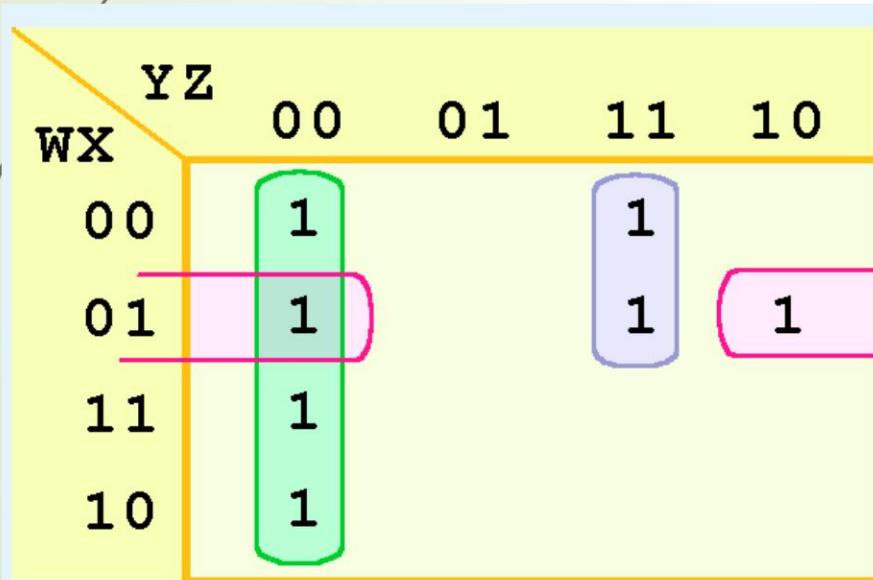
$$F(W, X, Y, Z) = \bar{W}\bar{Y} + \bar{X}\bar{Z} + \bar{W}YZ$$



Dört Değişken İçin Kmap Basitleştirmesi

52

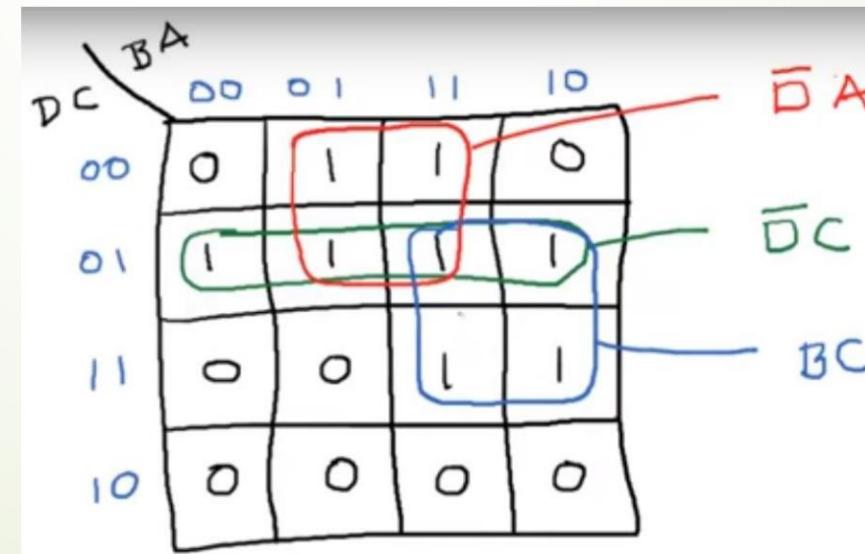
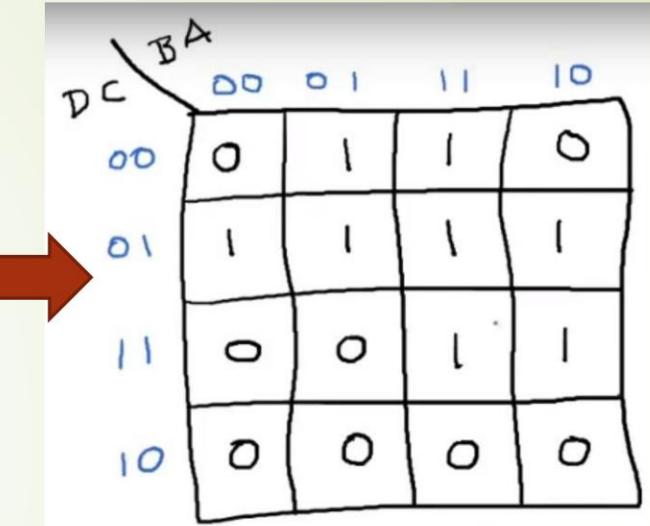
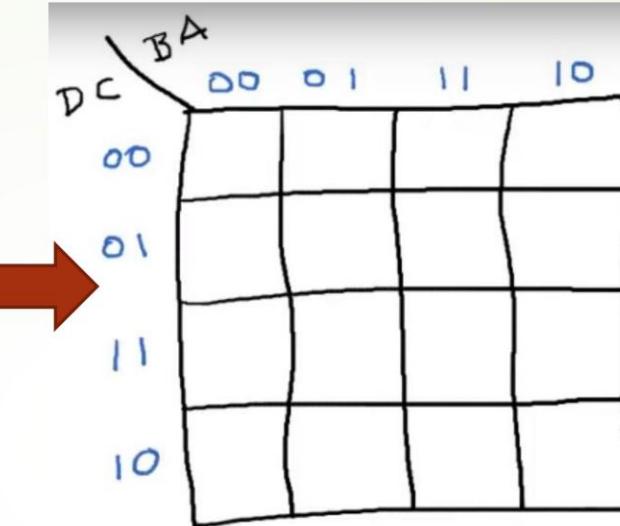
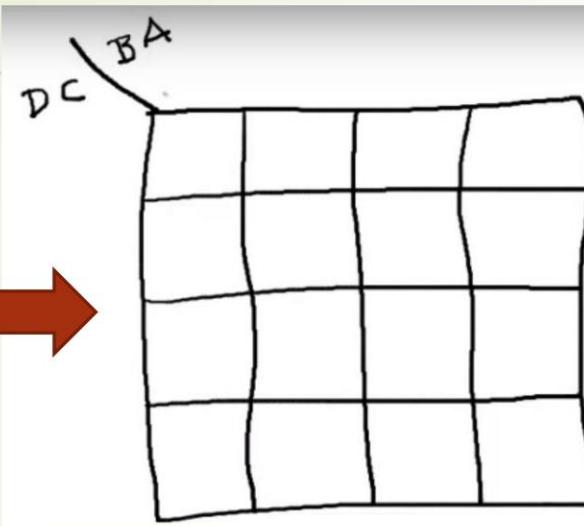
Grupları mümkün olduğunca büyük tutarak, bir Kmap içerisinde grupların nasıl seçileceği konusunda bir seçim yapmak mümkündür . Aşağıdaki gruplamalardan elde edilen (farklı) fonksiyonlar **mantıksal olarak eşdeğerdir.**



Dört Değişken İçin Kmap Basitleştirmesi

53

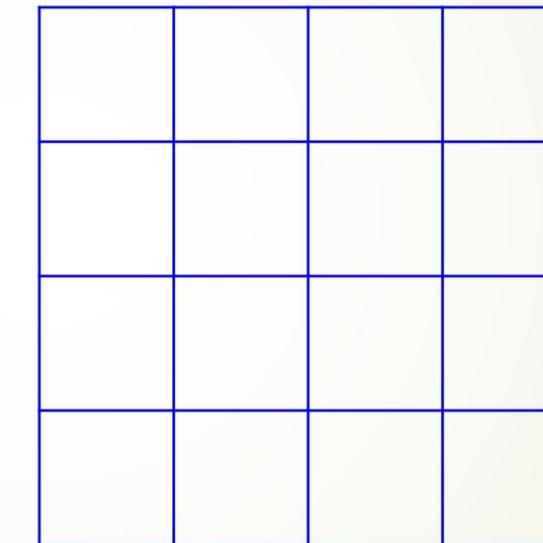
D	C	B	A	y
0	0	0	0	0
0	0	0	1	1
0	0	1	0	0
0	0	1	1	1
0	1	0	0	1
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	1
1	1	1	1	1



54

Dört Değişken İçin Kmap Basitleştirmesi

Türkçe: RS	T	Sen	F3	
0	0	0	0	0
0	0	0	1	1
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	0
1	0	0	1	1
1	0	1	0	0
1	0	1	1	0
1	1	0	0	1
1	1	0	1	0
1	1	1	0	1
1	1	1	1	1



55

Dört Değişken İçin Kmap Basitleştirmesi

Türkçe: RS	T	Sen	F3	
0	0	0	0	0
0	0	0	1	1
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	0
1	0	0	1	1
1	0	1	0	0
1	0	1	1	0
1	1	0	0	1
1	1	0	1	0
1	1	1	0	1
1	1	1	1	1

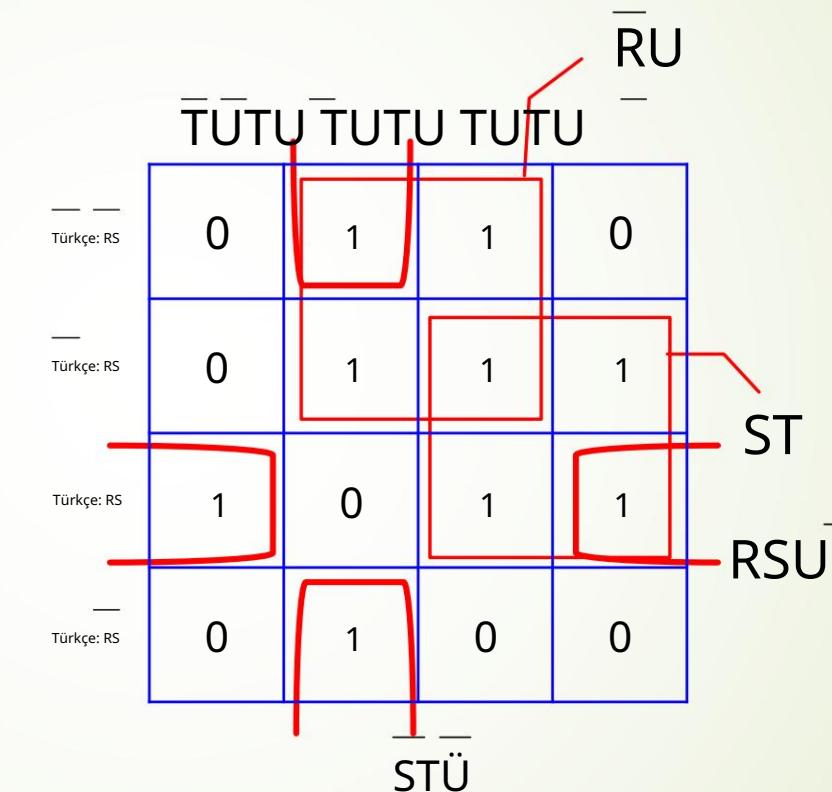
T̄ T U T̄ U T U T U

— —	0	1	1	0
—	0	1	1	1
Türkçe: RS	1	0	1	1
—	0	1	0	0

56

Dört Değişken İçin Kmap Basitleştirmesi

Türkçe: RS	TU	F3		
0	0	0	0	0
0	0	0	1	1
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	0
1	0	0	1	1
1	0	1	0	0
1	0	1	1	0
1	1	0	0	1
1	1	0	1	0
1	1	1	0	1
1	1	1	1	1



$$F3 = \overline{R} \overline{S}U + \overline{S}T \overline{U} + \overline{R}U + ST$$

Başlıca etkiler

Birincil **çıkarım**, haritadaki **mümkün** olan en fazla sayıda bitişik karenin birleştirilmesiyle elde edilen bir **ürün terimidir** .

Bir fonksiyonun birincil sonuçları şu şekilde elde edilebilir: mümkün olan tüm maksimum sayıda kareyi birleştirerek harita

Bu, bir haritada tek bir 1'in, diğer 1'lere bitişik değilse birincil bir anlamı temsil ettiği anlamına gelir.

Bir karedeki bir minterm **yalnızca bir asal sayı tarafından kapsanıyorsa ima edilen**, birincil ima edilenin esas olduğu söylenir .

Bir haritada bitişik kareleri seçerken, (1) fonksiyonun tüm mintermlerinin kapsandığından **emin** olmalıyız .

kareleri birleştir,

(2) ifadedeki terimlerin sayısı **en aza indirilir** ve (3) **gereksiz terimler yoktur** (yani, zaten var olan mintermler)
(diğer şartlar kapsamındadır).

Başlıca etkiler

İki bitişik 1, dört bitişik kareden oluşan bir grupta olmadıkları sürece bir asal sayı oluştururlar.

Eğer sekiz bitişik kareden oluşan bir grupta değilse , dört bitişik 1 bir asal sayı oluşturur , vb.

Haritadaki birincil çıkarımların tanımlanması, basitleştirilmiş bir ifade elde etmek için mevcut alternatiflerin belirlenmesine yardımcı olur

Her kombinasyon eşit derecede basitleştirilmiş bir sonuç üretебilir ifade.

Temel Olmayan ve Temel Birincil Etkiler

59

	AB	00	01	11	10
CD	00	0	1	1	0
	01	1	1	1	0
	11	1	0	0	0
	10	0	0	0	0

EACH of the coverings is a
PRIME IMPLICANT.

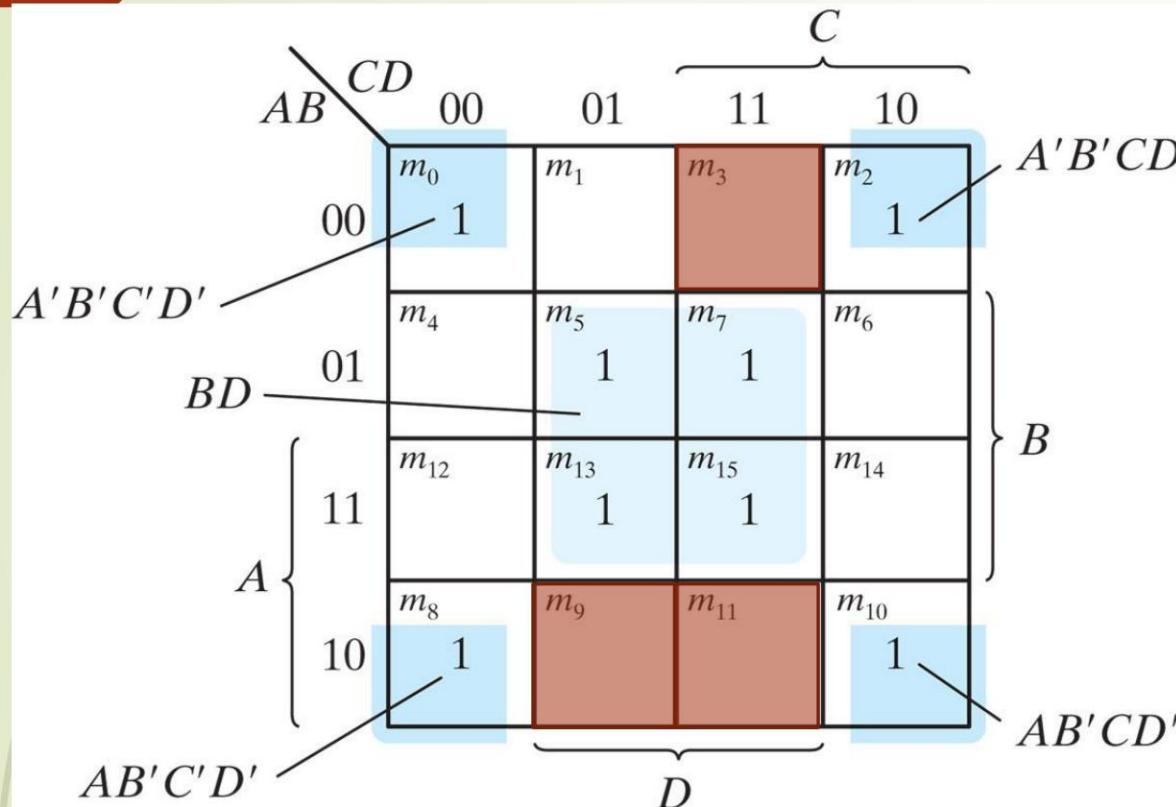
BC' , $A'C'D$, $A'B'D$

$$F(A,B,C,D) = BC' + A'B'D \quad (\text{minimum # of PIs})$$

Prime Implicant $A'C'D$ is a **NON-ESSENTIAL** prime implicant because its '1's are covered by other PIs. A PI is **ESSENTIAL** if it covers a MINTERM that cannot be covered by any other PI.

60

$F(A, B, C, D)$ için asal çıkarımlar = $(0, 2, 3, 5, 7, 8, 9, 10, 11, 13, 15)$

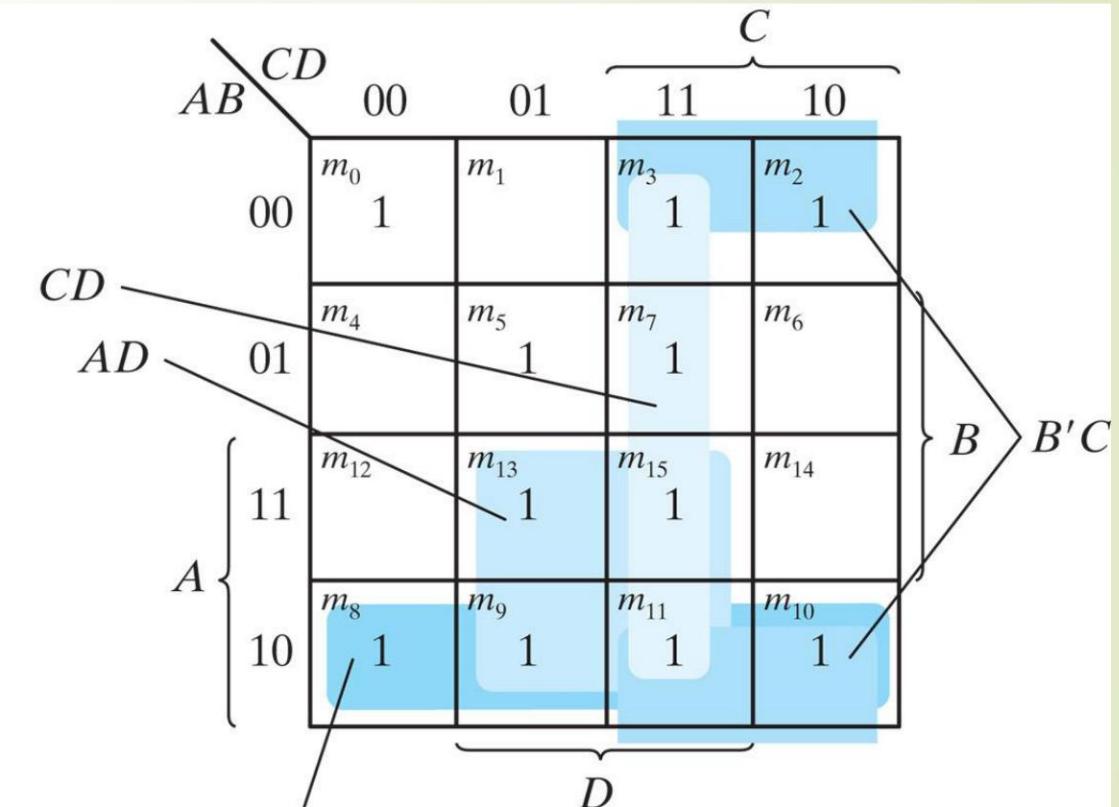


$$\text{Note: } A'B'C'D' + A'B'CD' = A'B'D'$$

$$AB'C'D' + AB'CD' = AB'D'$$

$$A'B'D' + AB'D' = B'D'$$

(a) Essential prime implicants
 BD and $B'D'$

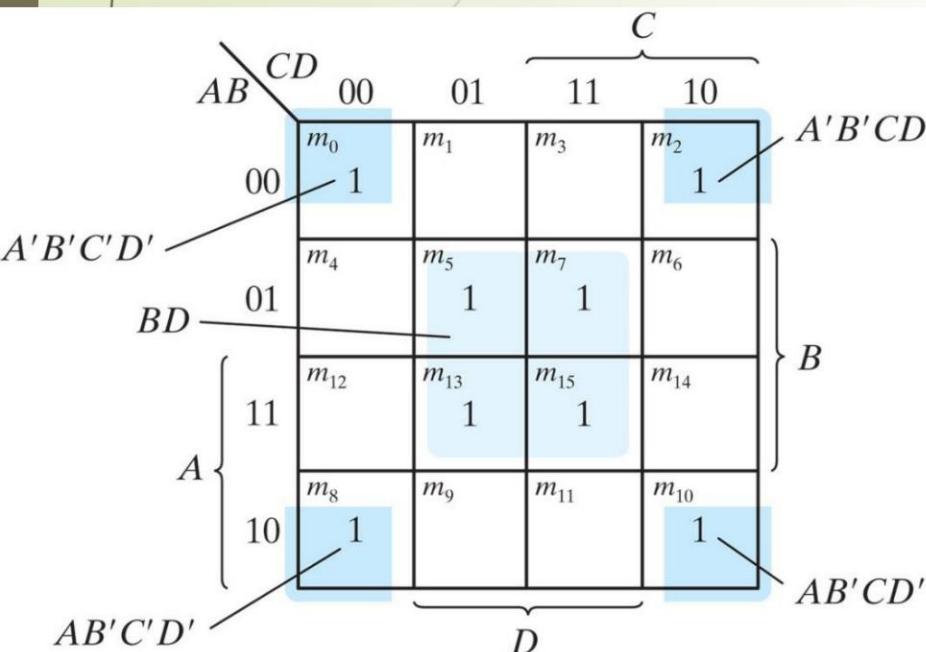


($m_3, m_9, \text{ ve } m_{11}$) dikkate alındı

(b) Prime implicants $CD, B'C$,
 AD , and AB'

61

$F(A, B, C, D) = (0, 2, 3, 5, 7, 8, 9, 10, 11, 13, 15)$ için asal çıkarımlar

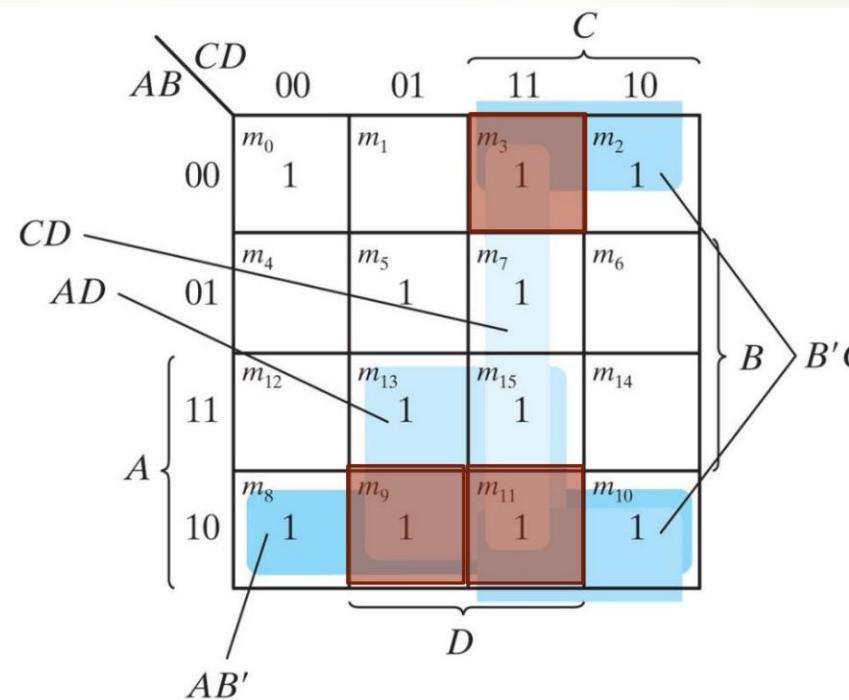


$$\text{Note: } A'B'C'D' + A'B'CD' = A'B'D'$$

$$AB'C'D' + AB'CD' = AB'D'$$

$$A'B'D' + AB'D' = B'D'$$

(a) Essential prime implicants
 BD and $B'D'$



(b) Prime implicants CD , $B'C$, AD , and AB'

Bu, tümünü gösterir

üç mintermin
($m_3, m_9,$

m_{11}) birincil
çıkarımlarıla
kapsanabilir .

$$= BD + B'D' + CD + AD$$

$$= BD + B'D' + CD + AB'$$

$$= BD + B'D' + B'C + AD$$

$$= BD + B'D' + B'C + AB'$$

Başlıca etkiler (Devamı)

Haritadan basitleştirilmiş ifadeyi bulma prosedürü, öncelikle tüm temel birincil çıkarımıları belirlememizi gerektirir

Basitleştirilmiş ifade, tüm temel asal çarpanların mantıksal toplamından **ve** temel asal çarpanlar tarafından kapsanmayan kalan mintermleri kapsamak için ihtiyaç duyulabilecek diğer asal çarpanlardan elde edilir.
ima edenler

Her kombinasyon eşit derecede basitleştirilmiş bir sonuç üretебilir ifade.

Beş ve Altı Değişkenli Haritalar

Değişken sayısı arttığında, kare sayısı aşırı hale gelir ve bitişik kareleri birleştirme geometrisi daha karmaşık hale gelir.

Dörtten fazla değişkene sahip haritaların kullanımı zordur, bu nedenle yazılım kullanmamız gereklidir.

Beş ve altı değişkenli Haritalar

Sol taraftaki dört değişkenli harita, 16 kareyi temsil eder.

$A=0$ ve diğer dört değişkenli harita $A=1$ olan kareleri temsil eder.

Ayrıca $A=0$ haritasındaki her kare, $A=1$ haritasındaki **ilgili kareye** bitişiktir .

$A = 0$					
DE		D			
BC		00	01	11	10
B	00	0	1	3	2
	01	4	5	7	6
	11	12	13	15	14
	10	8	9	11	10
E					

$A = 1$					
DE		D			
BC		00	01	11	10
B	00	16	17	19	18
	01	20	21	23	22
	11	28	29	31	30
	10	24	25	27	26
E					

65

Altı değişkenli k-haritası

64 hücre

Video

TOPLAMLARIN ÇARPIMI BASİTLEŞTİRİLMESİ

Önceki örneklerin tümü [ürün-toplamında](#) ifade edilmiştir
biçim

Haritanın karelerine yerleştirilen 1'ler, mintermeleri temsil eder.
fonksiyon

Bir fonksiyonun tamamlayıcısı haritada 1'lerle işaretlenmemiş karelerle gösterilmektedir.

Boş kareleri 0'larla işaretler ve bunları geçerli bitişik karelere birleştirirsek, basitleştirilmiş bir ürün toplamı elde ederiz
fonksiyonun tamamlayıcısının ifadesi (yani, F' nin).

F' nin tamamlayıcısı bize F fonksiyonunu toplamların çarpımı biçiminde geri verir
(DeMorgan teoreminin bir sonucu).

Genelleştirilmiş DeMorgan teoremi nedeniyle, elde edilen fonksiyon otomatik olarak
[toplamların çarpımı biçimindedir](#)

67

Örnek 3.7 Toplamların çarpımlarının

EXAMPLE 3.7 basitleştirilmesi

Simplify the following Boolean function into (a) sum-of-products form and (b) product-of-sums form:

$$F(A, B, C, D) = \Sigma(0, 1, 2, 5, 8, 9, 10)$$

68

Örnek 3.7 Toplamların çarpımlarının basitleştirilmesi

$$F(A, B, C, D) = (0, 1, 2, 5, 8, 9, 10)$$

Ürünlerin Toplami formu

$$F = B'D' + B'C' + A'C'D$$

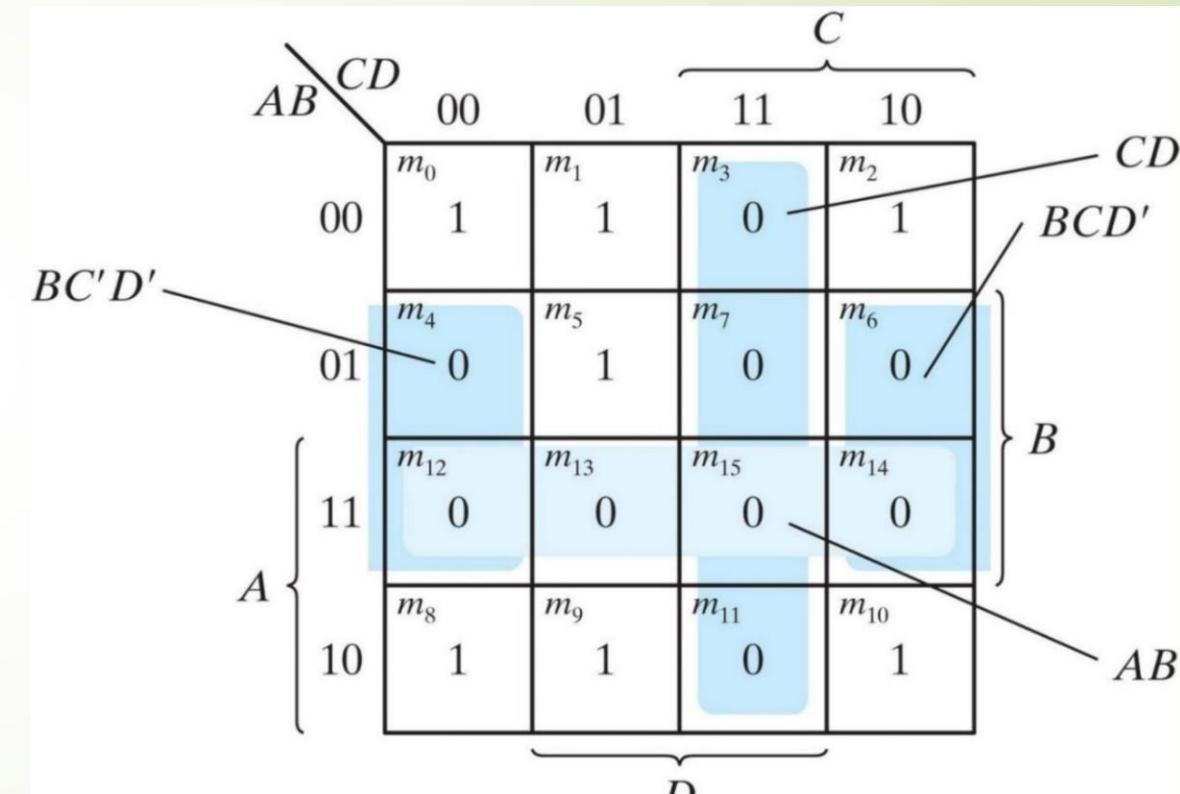
$$F' = AB + CD + BD'$$

DeMorgan teoremini uygulamak

ikiliyi alarak ve her bir
sabiti tamamlayarak
basitleştirilmiş fonksiyonu elde ederiz

Toplamların çarpımı biçiminde:

$$F = (A' + B') (C' + D') (B' + D)$$

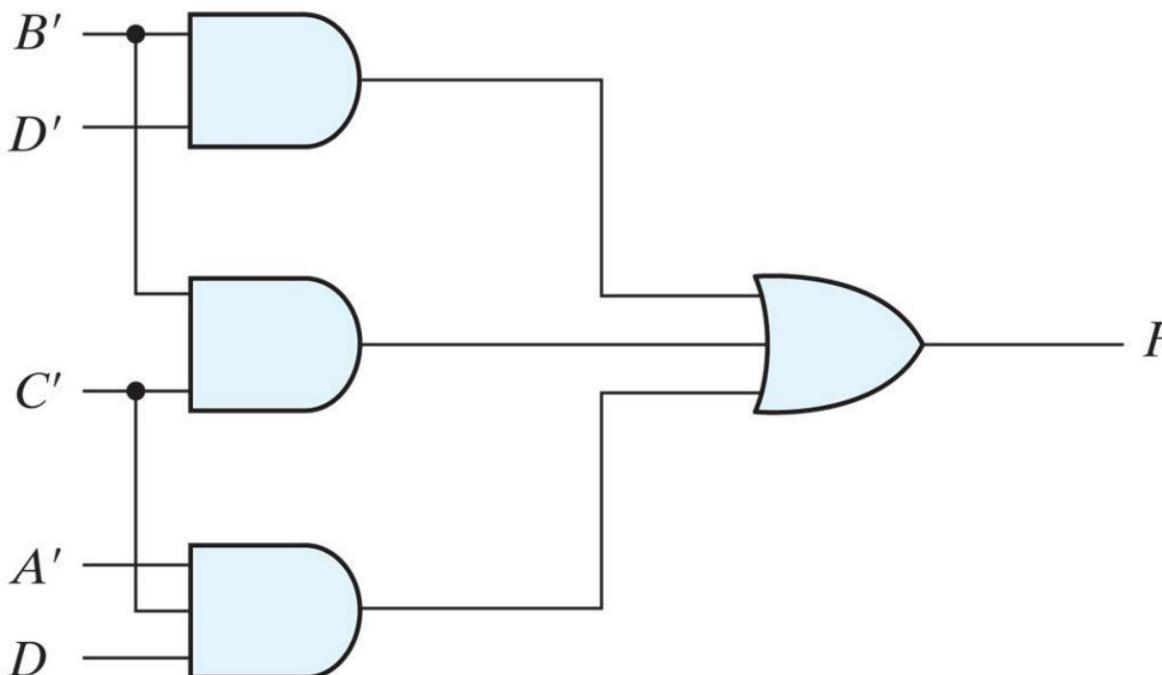


$$\text{Note: } BC'D' + BCD' = BD'$$

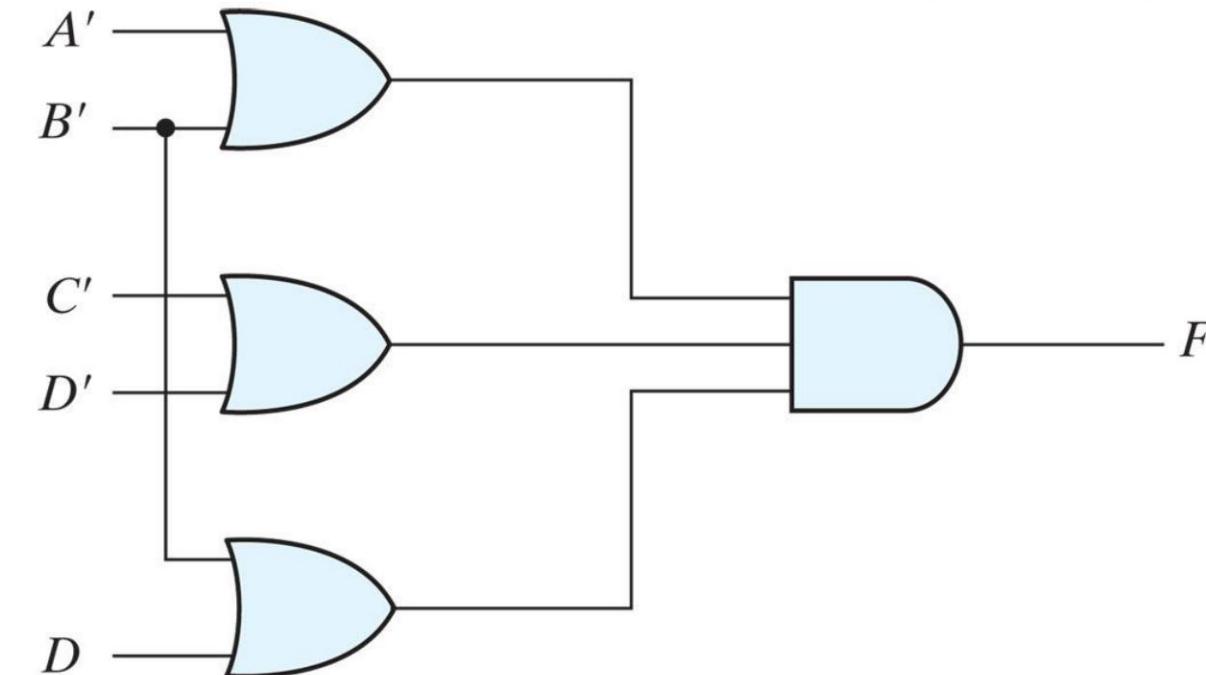
Copyright ©2013 Pearson Education, publishing as Prentice Hall

69

Örn.3.7'deki fonksiyonun kapı uygulamaları



$$(a) F = B'D' + B'C' + A'C'D$$



$$(b) F = (A' + B') (C' + D') (B' + D)$$

Toplamların çarpımlarının basitleştirilmesi

Örneğin, Tablo 3.1'deki F fonksiyonunu tanımlayan doğruluk tablosunu ele alalım.

Mintermlerin toplamı biçiminde bu fonksiyon şu şekilde ifade edilir:

$$F(x, y, z) = (1, 3, 4, 6)$$

Maksimum terimlerin çarpımı biçiminde şu şekilde ifade edilir:

$$F(x, y, z) = (0, 2, 5, 7)$$

Table 3.1
Truth Table of Function F

x	y	z	F
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

71

Tablo 3.1'deki fonksiyona ait harita

Ürünlerin toplamı için 1'leri birleştirerek şunu elde ederiz:

$$F = x'z + xz'$$

Toplamların çarpımı için 0'ları birleştirerek şunu elde ederiz:

$$F' = xz + x'z'$$

F 'nin tamamlayıcısını alarak, toplamların çarpımı
formunda basitleştirilmiş fonksiyonu elde ederiz :

$$F = (x' + z')(x + z)$$

	00	01	11	10
0	m_0 0	m_1 1	m_3 1	m_2 0
1	m_4 1	m_5 0	m_7 0	m_6 1

Toplamların çarpım formunun haritaya girilmesi

Toplamların çarpımı biçiminde ifade edilen bir fonksiyonu haritaya girmek için, 0'larla işaretlenecek kareleri bulmak için fonksiyonun tamamlayıcısını kullanın .

Örneğin:

$$F = (A' + B' + C')(B + D)$$

Öncelikle tamamlayıcısını alalım:

$$F' = ABC + B'D'$$

Daha sonra kareleri temsil eden 0'ları işaretleyin
F'nin mintermleri . Kalan kareler 1'lerle işaretlenmiştir.

3.5 Önemsiz Koşullar

Gerçek devrelerin her olası giriş için tanımlanmış bir çıktıya sahip olması gerekmektedir.
(Pratikte, bazı uygulamalarda fonksiyon değişkenlerin belirli kombinasyonları
için belirtilemez.)

Örneğin, bazı hesap makinesi ekranları 7 segmentli LED'lerden oluşur. Bu LED'ler 2 7 -1
desenini görüntüleyebilir, ancak bunlardan sadece on tanesi kullanışlıdır.

Belirtilmemiş bazı girdi çıktılarına sahip işlevler
kombinasyonlara **eksik belirtilen fonksiyonlar** denir

Eğer bir devre, belirli bir girdi kümесinin asla gerçekleşmeyeceği şekilde
tasarlanmışsa, bu girdi kümese umursamaz koşulu adını veririz.

K-haritası devre basitleştirmesinde bize çok yardımcı olurlar ve bir haritada Boole
ifadesinin daha da basitleştirilmesini sağlamak için önemsiz koşullar
kullanılabilir .

3.5 Önemsiz Koşullar

Bir Boole fonksiyonuyla ilişkili mintermlerin mantıksal toplamı, fonksiyonun 1'e eşit olduğu koşulları belirtir.

Fonksiyon, geri kalan mintermler için 0'a eşittir. Bu koşul çifti,
aşağıdakilerin tüm kombinasyonlarının
Fonksiyonun değişkenlerine ait değerler geçerlidir.

Uygulamada, bazı uygulamalarda fonksiyon belirtilmemiştir
değişkenlerin belirli kombinasyonları.

Örn: Ondalık basamaklar için dört bitlik ikili kod, kullanılmayan ve dolayısıyla belirtilmemiş olarak kabul edilen altı kombinasyona sahiptir .

Bazı girdi kombinasyonları için belirtilmemiş çıktıları olan işlevlere **eksik belirtilmiş işlevler** denir

Belirtilmemiş mintermler yaratın

Önemsiz koşulu 1'lerden ve 0'lardan ayırt etmek için bir **X** kullanılır

3.5 Önemsiz Koşullar

Bir Kmap'te, önemsiz bir koşul , aşağıda gösterildiği gibi önemsiz girdiler için minterm(ler)in hücresindeki bir X ile tanımlanır.

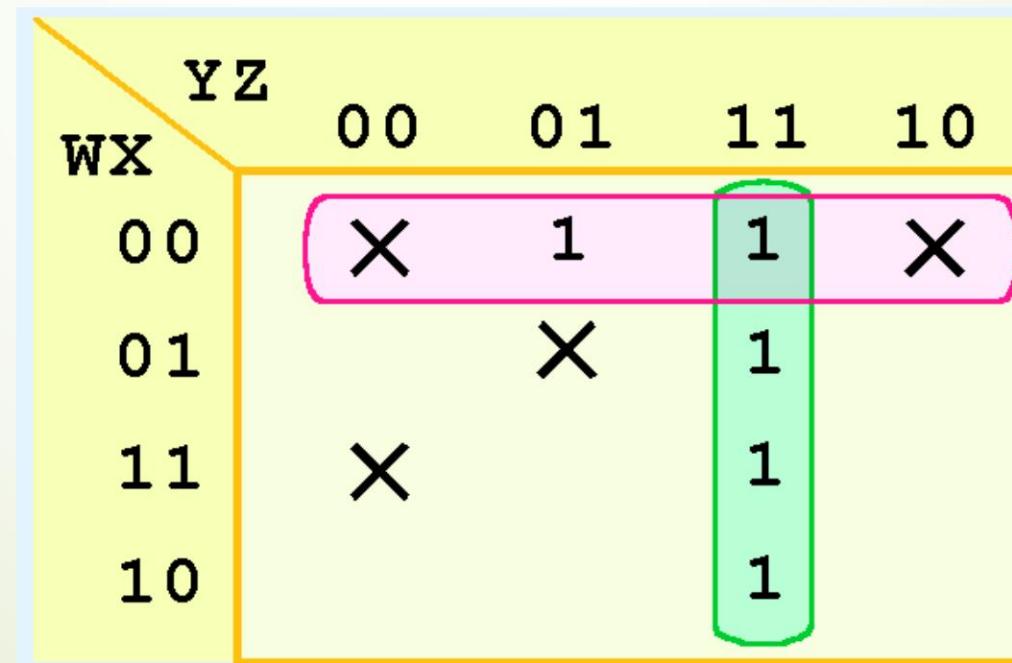
Basitleştirmeyi yaparken, gruplarımıza oluştururken **X'leri** dahil etmek veya yok saymak konusunda serbestiz .

YZ	00	01	11	10
WX				
00	X	1	1	X
01		X	1	
11	X		1	
10			1	

3.5 Önemsiz Koşullar

Aşağıdaki Kmap'teki bir grupta,
işlev:

$$F(W, X, Y, Z) = \overline{W}\overline{Y} + YZ$$



3.5 Önemsiz Koşullar

Farklı bir gruplama bize şu fonksiyonu verir:

$$F(W, X, Y, Z) = \overline{WZ} + YZ$$

WX	YZ	00	01	11	10
00	X	1		1	X
01	X		X	1	
11	X			1	
10				1	1

78

3.5 Önemsiz Koşullar

Şunun doğruluk tablosu:

$$F(W, X, Y, Z) = \overline{W}\overline{Y} + YZ$$



Aşağıdaki doğruluk tablosundan **farklıdır**:

$$F(W, X, Y, Z) = \overline{W}\overline{Z} + YZ$$

Ancak, farklılık gösterdikleri değerler, umursamadığımız koşullara sahip girdilerdir.

		YZ	00	01	11	10
		WX	00	01	11	10
00	00	X	1	1	X	
				X	1	
01	01		X	1		
					X	
11	11	X		1		
					X	
10	10				1	
						1

		YZ	00	01	11	10
		WX	00	01	11	10
00	00	X	1	1	X	
				X	1	
01	01		X	1		
					X	
11	11	X		1		
					X	
10	10				1	
						1

3.6 NAND ve NOR Uygulaması

Dijital devreler sıkılıkla AND ve OR kapılarından ziyade NAND veya NOR kapılarıyla oluşturulur. NAND ve NOR kapılarının üretimi daha kolaydır

elektronik bileşenlerdir ve tüm IC dijital mantık ailelerinde kullanılan temel kapılardır .

İki seviyeli formlar

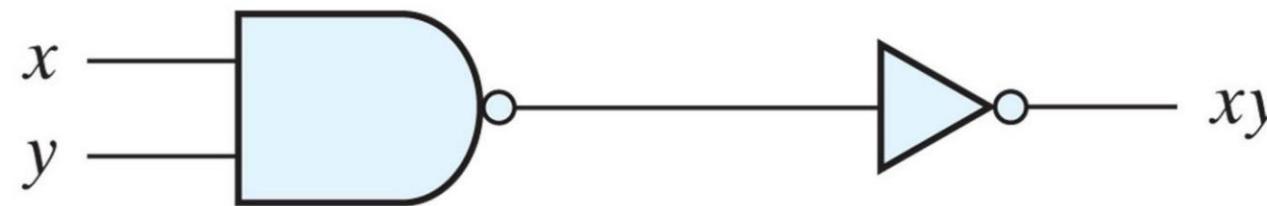
Çok seviyeli formlar

NAND kapılarıyla mantık işlemleri

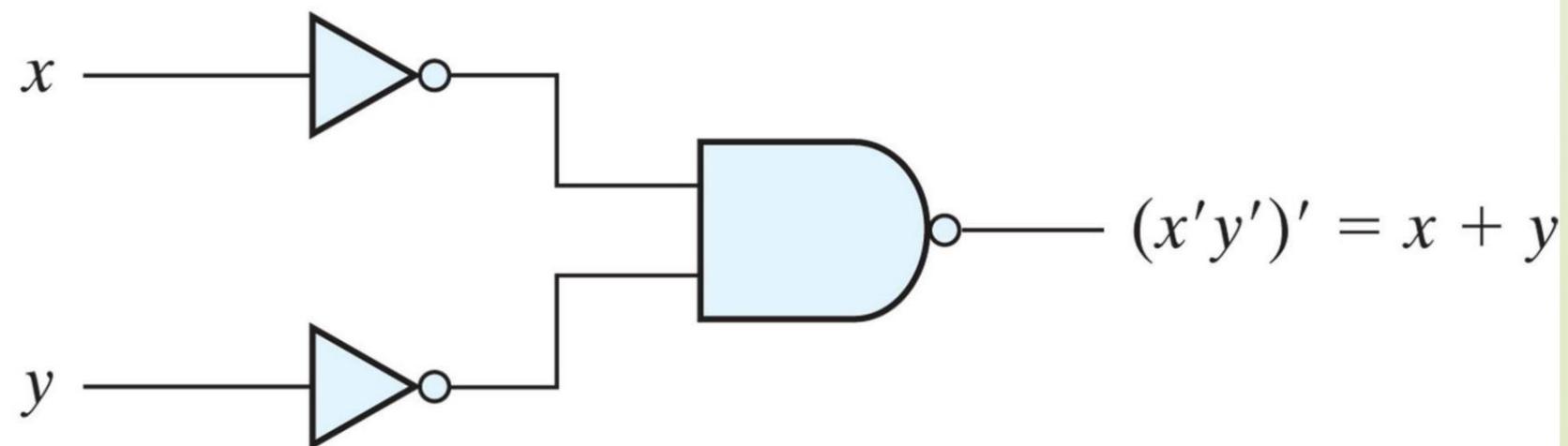
Inverter



AND

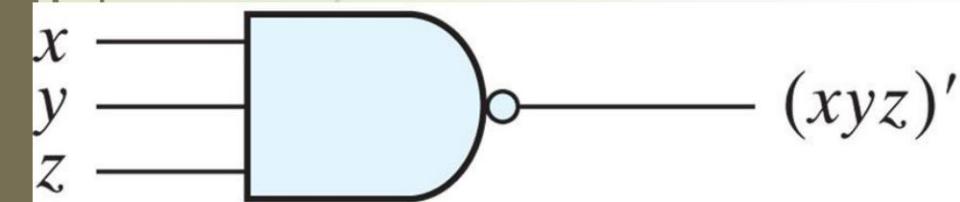


OR

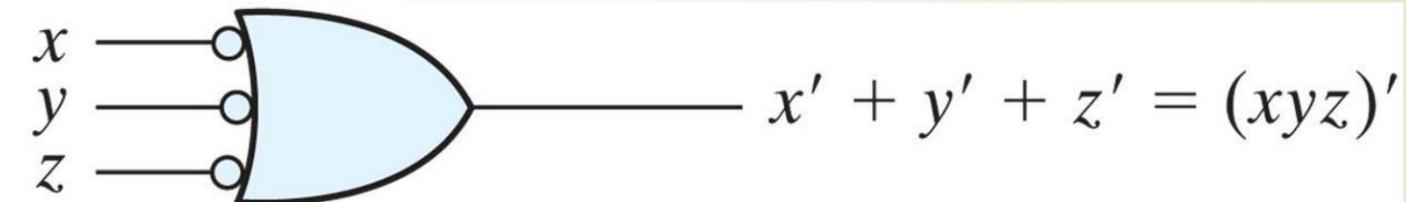


NAND kapılarıyla mantık işlemleri

Üç girişli bir NAND kapısı için iki grafik sembolü

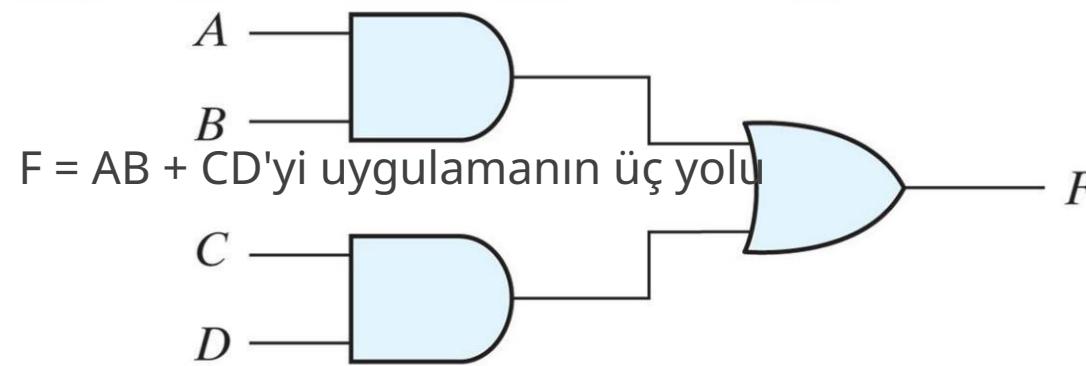


(a) AND-invert

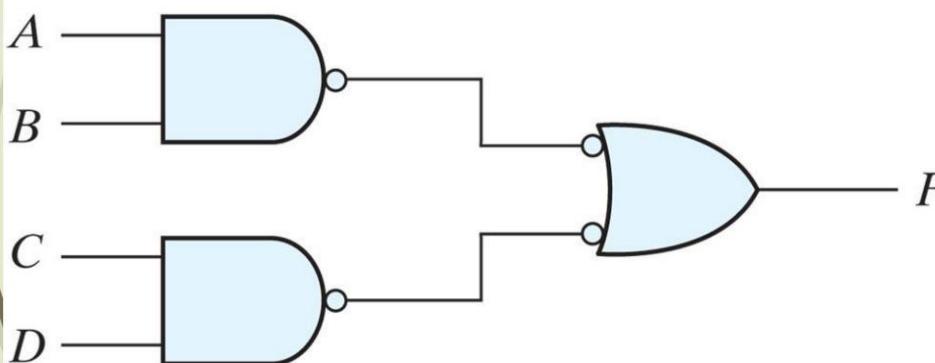


(b) Invert-OR

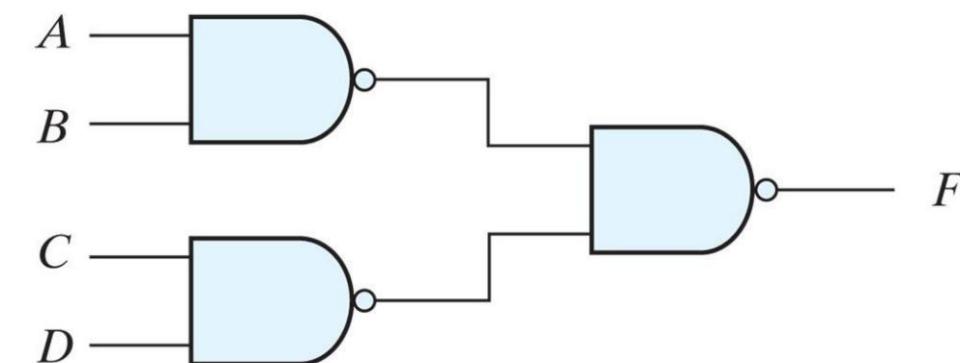
İki seviyeli formlar



(a)



(b)



(c)

Elde etme prosedürü Boolean fonksiyonundan mantık diyagramı

Fonksiyonu sadeleştirin ve çarpımların toplamı biçiminde ifade edin.

En az iki sabite sahip ifadenin her ürün terimi için bir NAND kapısı çizin . Her NAND kapısının girdileri terimin sabitleridir. Bu prosedür **birinci seviye kapılardan** oluşan bir grup üretir.

AND-invert veya invert-OR kullanarak tek bir kapı çizin
İkinci seviyedeki grafik simbol, girişlerin birinci seviyedeki kapıların çıkışlarından gelmesiyle oluşur.

Tek bir sabite sahip bir terim, birinci seviyede bir invertör gerektirir.
Ancak tekil literal tamamlayıcı ise, doğrudan ikinci seviye NAND kapısının bir girişine bağlanabilir.

NAND kapısı örneği

EXAMPLE 3.9

Implement the following Boolean function with NAND gates:

$$F(x, y, z) = (1, 2, 3, 4, 5, 7)$$

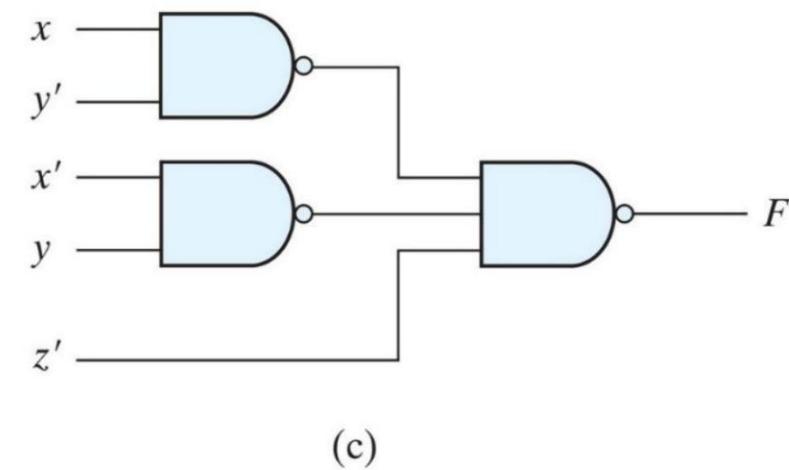
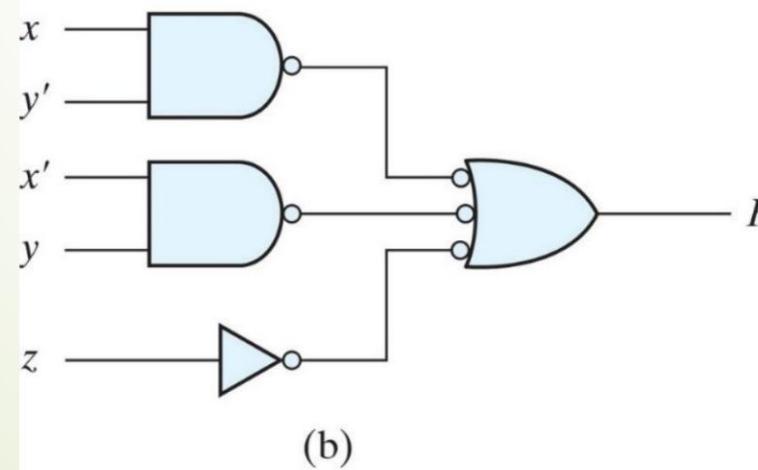
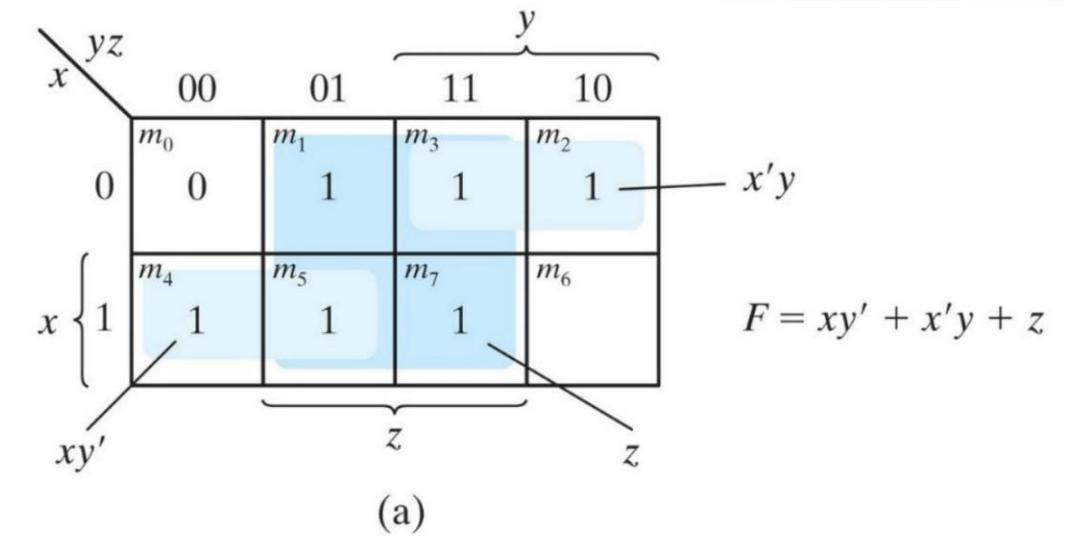
İlk adım, fonksiyonu ürünlerin toplamı biçimine sadeleştirmektir .

$$F = xy' + x'y + z$$

85

NAND kapısı örneği (3.9)

$$F = xy' + x'y + z$$

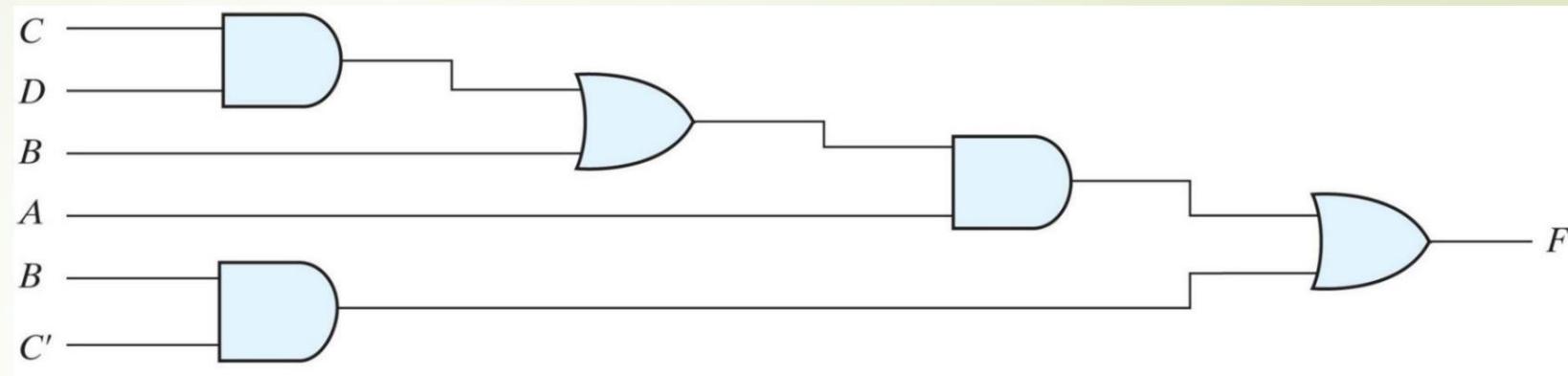


Çok seviyeli formlar

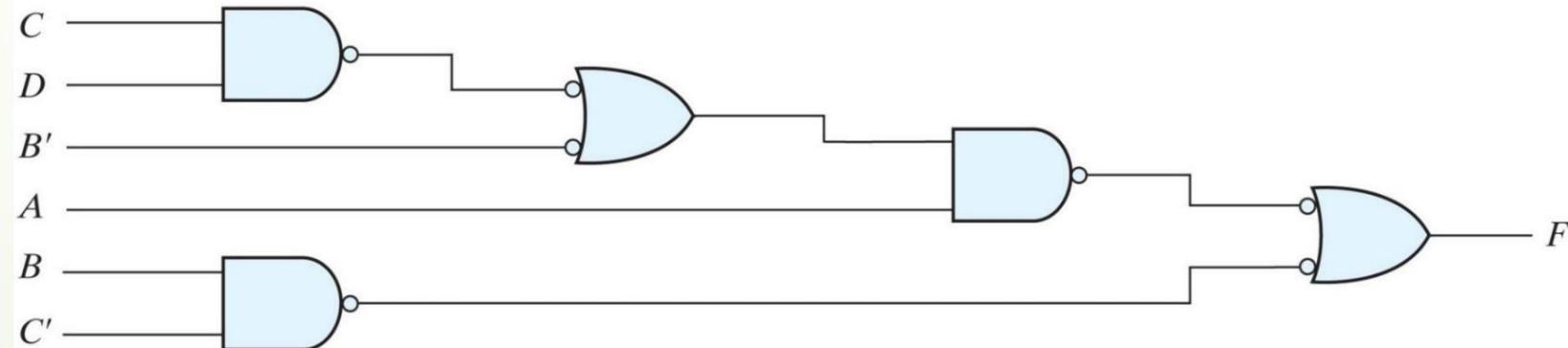
$$F = A(CD + B) + BC'$$

Prosedür, her VE kapısını bir VE-ters grafik sembolüne ve her VEYA kapısını bir ters-VEYA grafik sembolüne dönüştürmektedir.

Aynı çizgide iki baloncuk



(a) AND-OR gates



(b) NAND gates

Çok seviyeli bir yapıyı dönüştürmenin genel prosedürü

VE-VEYA diyagramını karma gösterim kullanarak tüm-NAND diyagramına dönüştürme

Tüm AND kapılarını AND-ters grafik sembollerile NAND kapılarını dönüştürün

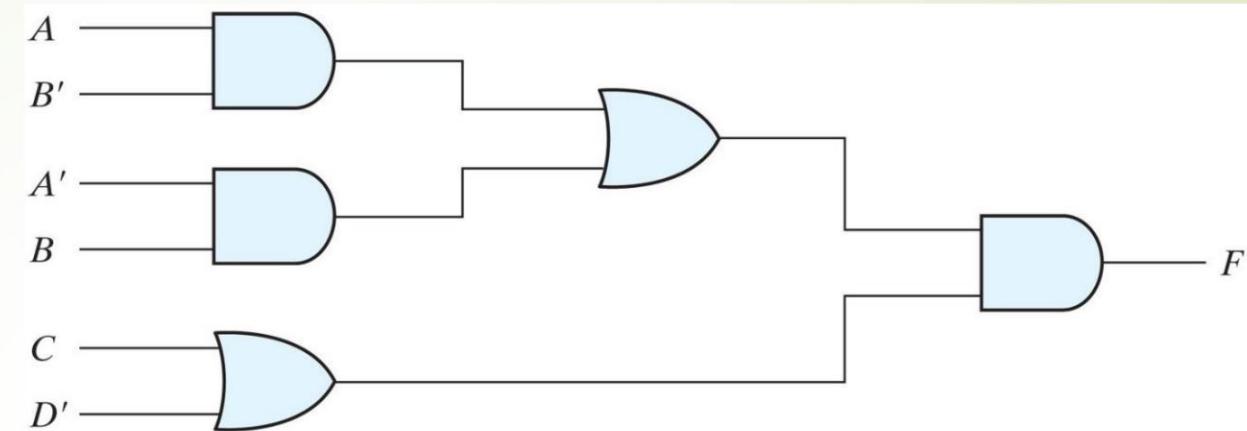
Tüm VEYA kapılarını ters VEYA grafik sembollerile NAND kapılarını dönüştürün

Diyagramdaki tüm baloncukları kontrol edin. Aynı çizgi boyunca başka bir küçük daire tarafından telafi edilmeyen her baloncuk için bir invertör (tek girişli NAND kapısı) ekleyin veya giriş sabitini tamamlayın.

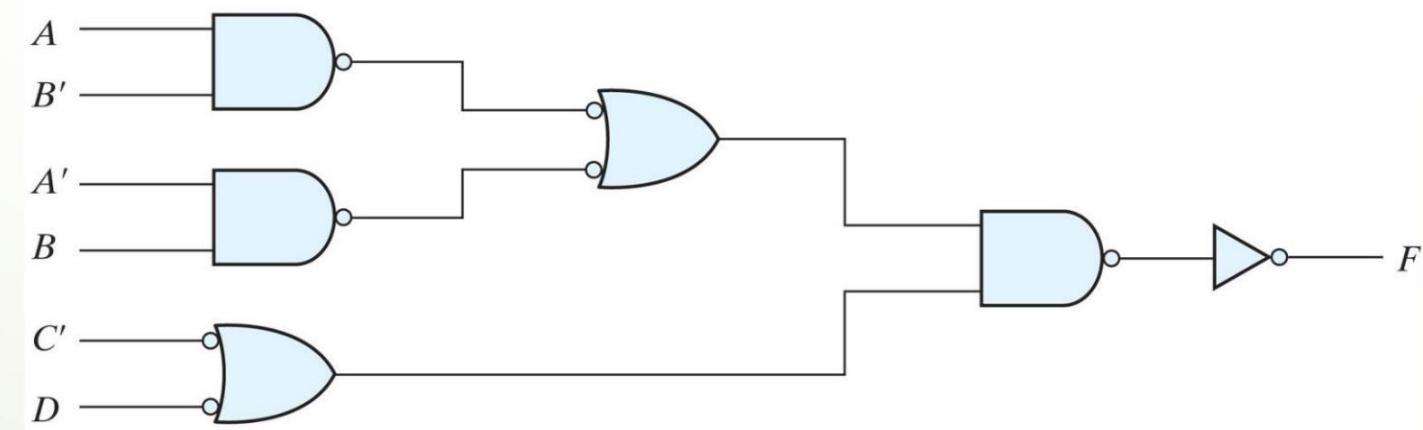
88

AND-OR diyagramını NAND'a dönüştürme

$$F = (AB' + A'B)(C + D')$$



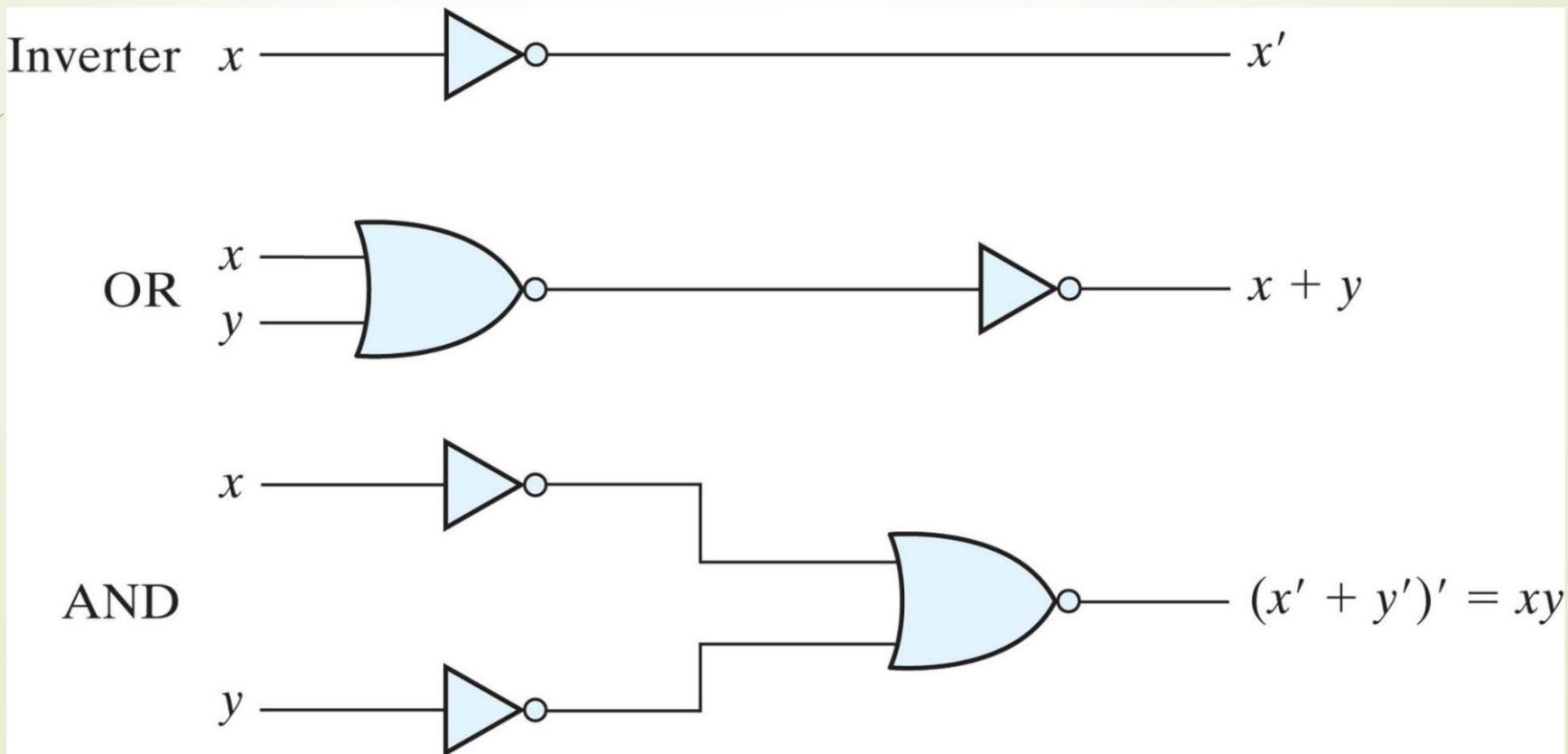
(a) AND-OR gates



(b) NAND gates

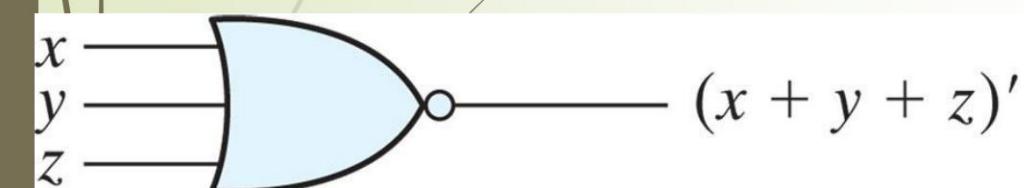
NOR kapılarıyla mantıksal işlemler

NOR işlemi, NAND işleminin ikilidir

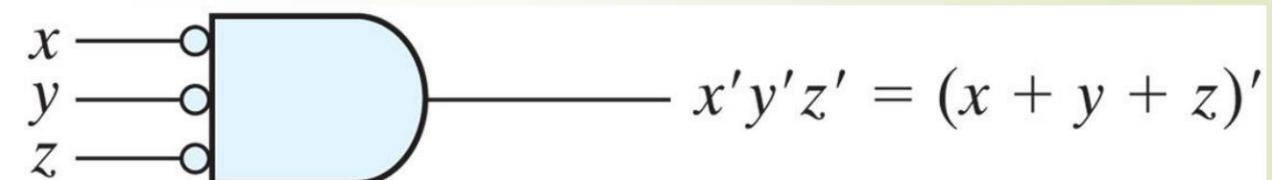


NOR kapılarıyla mantıksal işlemler

NOR kapısı için iki grafik simbolü



(a) OR-invert

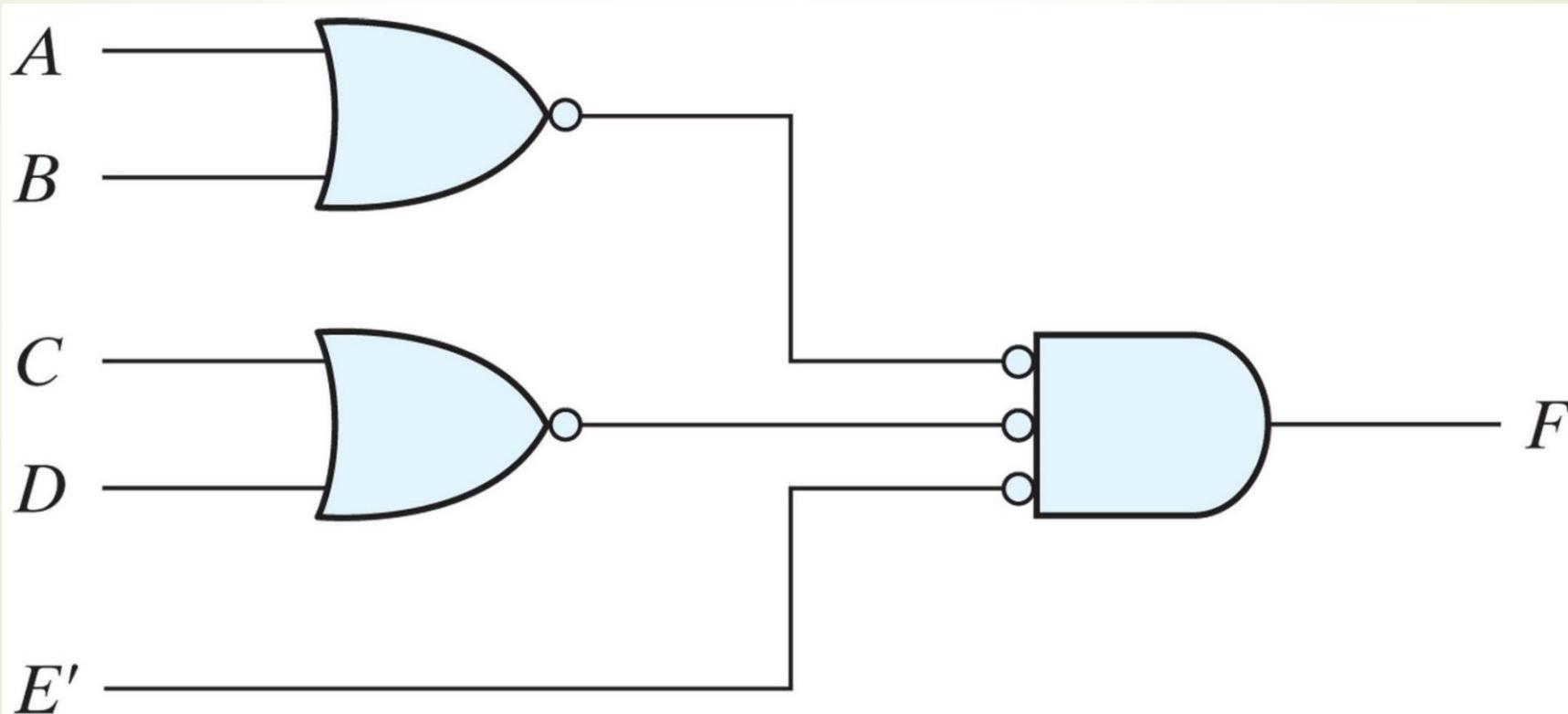


(b) Invert-AND

91

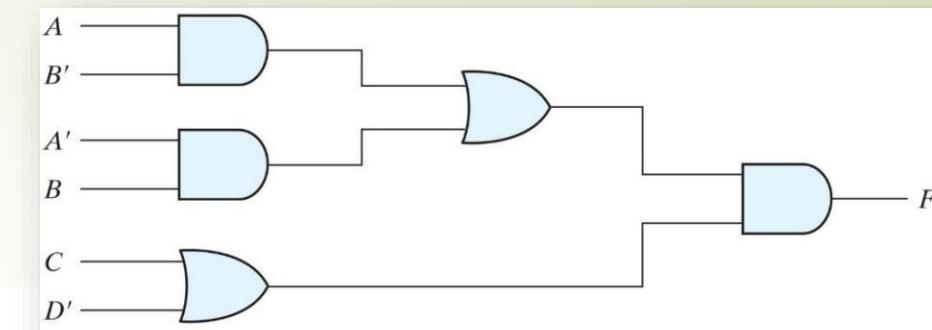
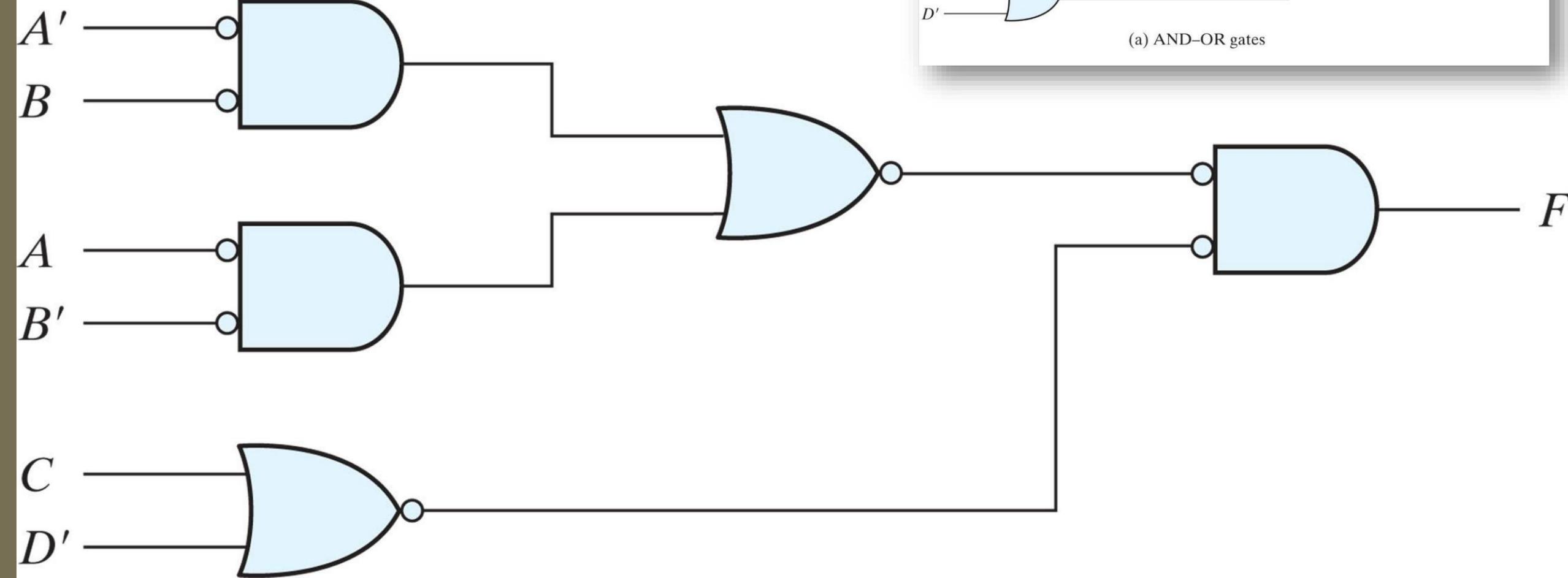
Örnek NOR uygulaması

$$F = (A + B)(C + D)E$$



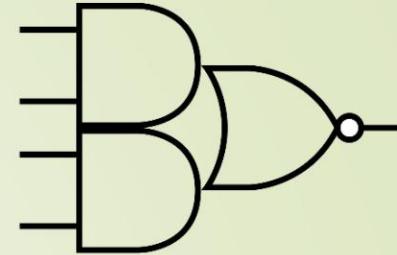
$F = (AB' + A'B)(C + D)'$ nin NOR kapılarıyla uygulanması

92

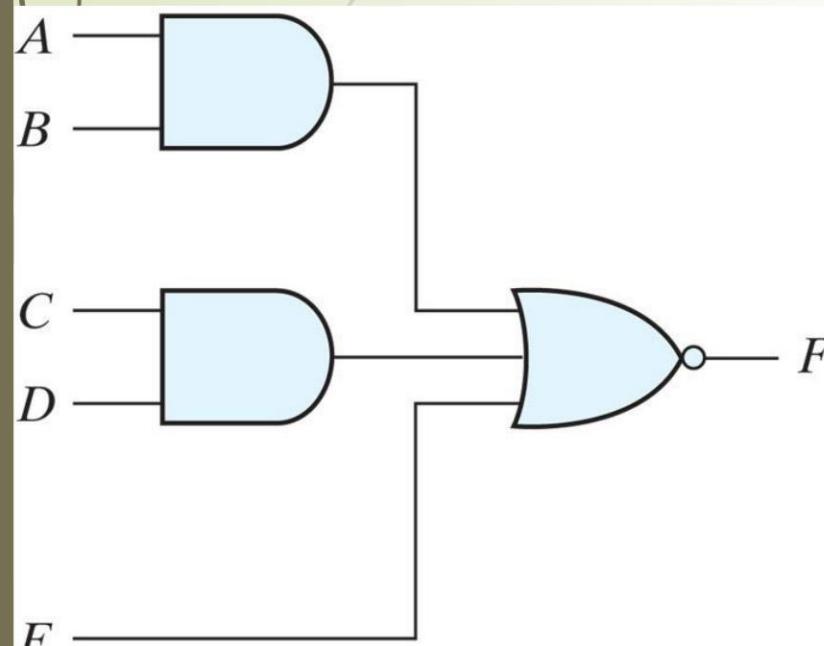


(a) AND-OR gates

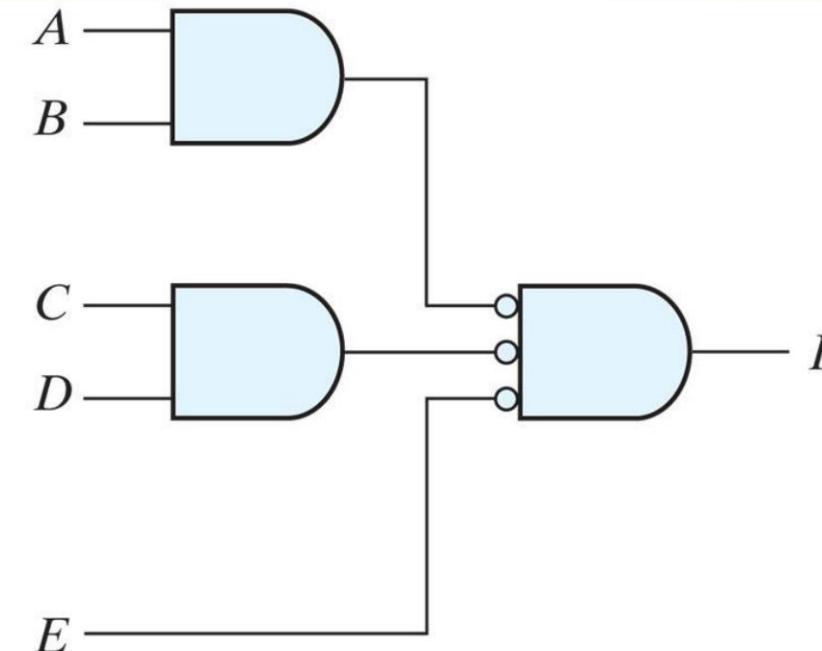
VE-VEYA-TERS Uygulamasi



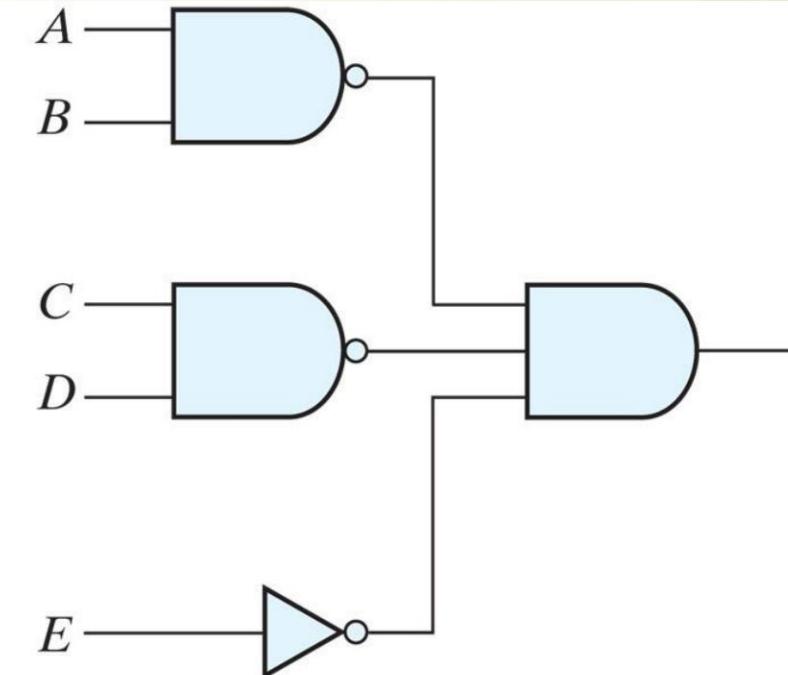
$$F = (AB + CD + E)'$$



(a) AND-NOR



(b) AND-NOR



(c) NAND-AND

94

Tek Fonksiyon

Üç değişkenli özel VEYA fonksiyonu, yalnızca bir değişken 1'e eşitse veya üç değişken de 1'e eşitse 1'e eşittir.

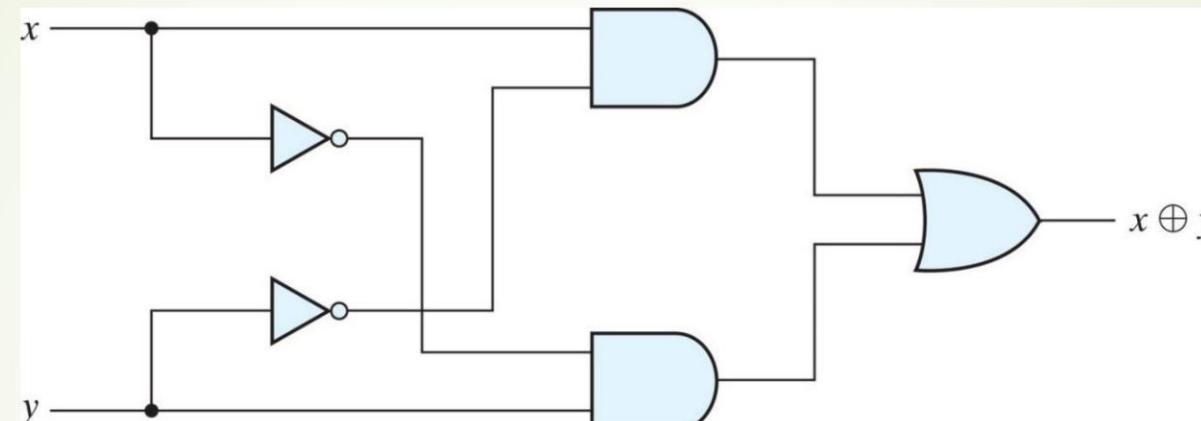
İki değişkenli durumda, yalnızca bir değişkenin 1'e eşit olması gereklidir, üç veya daha fazla değişken durumunda, tek sayıda değişkenin 1'e eşit olması gereklidir.

Bu nedenle, çok değişkenli özel VEYA işlemi **tek bir fonksiyon** olarak tanımlanır

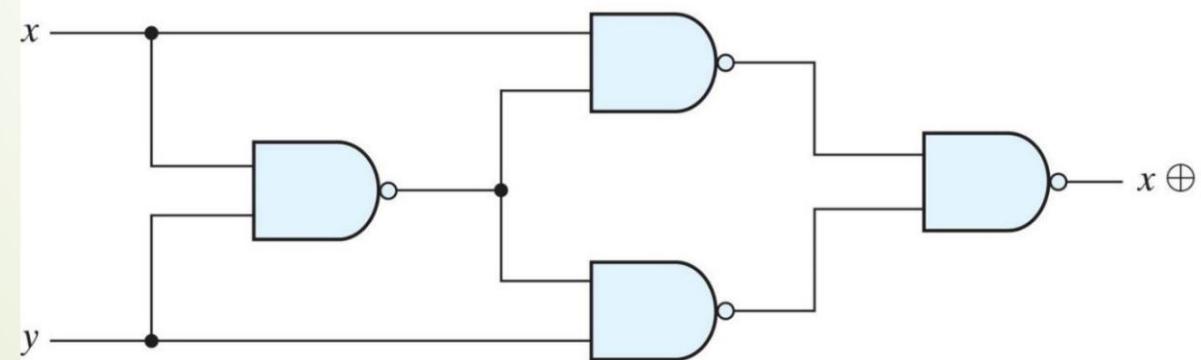
INPUTS		OUTPUTS
A	B	$Y = A \oplus B$
0	0	0
0	1	1
1	0	1
1	1	0

3-input XOR gate			
A	B	C	Output
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

Özel OR uygulamaları



(a) Exclusive-OR with AND-OR-NOT gates



(b) Exclusive-OR with NAND gates

XOR kullanan Tek ve Çift fonksiyonları

Şimdi dört değişkenli özel VEYA işlemini ele alalım

$$A \oplus B \oplus C \oplus D = (AB' + A'B) \oplus (CD' + C'D)$$

$$= (AB' + A'B)(CD + C'D') + (AB + A'B')(CD' + C'D)$$

$$= \Sigma(1, 2, 4, 7, 8, 11, 13, 14)$$

97

XOR kullanan Tek ve Çift fonksiyonları

Karnaugh map for an odd function $F = A \oplus B \oplus C \oplus D$:

		C			
		00	01	11	10
$A\backslash B$	00	m_0	m_1	m_3	m_2
01	m_4	m_5	m_7	m_6	
11	m_{12}	m_{13}	m_{15}	m_{14}	
10	m_8	m_9	m_{11}	m_{10}	

Variables: AB , CD . Minterms: $m_0, m_1, m_3, m_2, m_4, m_5, m_7, m_6, m_{12}, m_{13}, m_{15}, m_{14}, m_8, m_9, m_{11}, m_{10}$.

(a) Odd function $F = A \oplus B \oplus C \oplus D$

Karnaugh map for an even function $F = (A \oplus B \oplus C \oplus D)'$:

		C			
		00	01	11	10
$A\backslash B$	00	m_0	m_1	m_3	m_2
01	m_4	m_5	m_7	m_6	
11	m_{12}	m_{13}	m_{15}	m_{14}	
10	m_8	m_9	m_{11}	m_{10}	

Variables: AB , CD . Minterms: $m_0, m_1, m_3, m_2, m_4, m_5, m_7, m_6, m_{12}, m_{13}, m_{15}, m_{14}, m_8, m_9, m_{11}, m_{10}$.

(b) Even function $F = (A \oplus B \oplus C \oplus D)'$

Parite Oluşturma ve Kontrol Etme

Exclusive-OR fonksiyonları hata tespiti ve düzeltme kodları gerektiren sistemlerde çok faydalıdır

İkili bilginin iletimi sırasında **hataları tespit etmek amacıyla bir parite biti** kullanılır

Bir parite biti, ikili bir veriye dahil edilen ekstra bir bittir.
1'lerin sayısını **tek** veya **çift** yapmak için mesaj

Parite Oluşturma ve Kontrol Etme

Eşlik biti de dahil olmak üzere mesaj iletilir ve ardından alıcı tarafta hatalar açısından kontrol edilir.

Parite Üreticisi

Vericide parite bitini üreten devreye parite üretici adı verilir .

Parite Denetleyicisi

Alıcıdaki pariteyi kontrol eden devreye parite denetleyicisi adı verilir .

Çift-Parite-Üretici Doğruluk Tablosu

P , üç değişkenli özel VEYA fonksiyonu olarak ifade edilebilir

$$P = x \oplus y \oplus z$$

Table 3.3
Even-Parity-Generator Truth Table

Three-Bit Message			Parity Bit
x	y	z	P
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

101

Hamming Hata Düzeltme Kodları

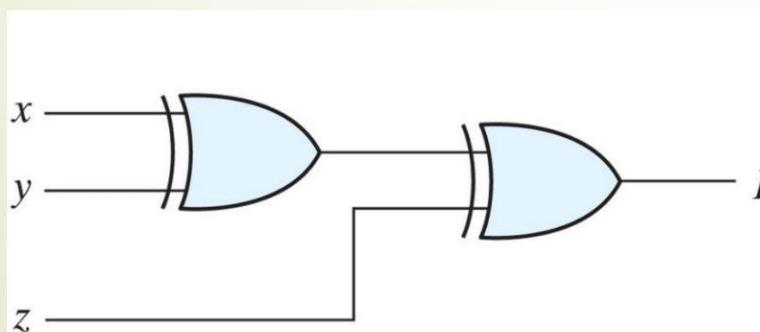
102

Bir parite üreteci ve denetleyicisinin mantık diyagramı

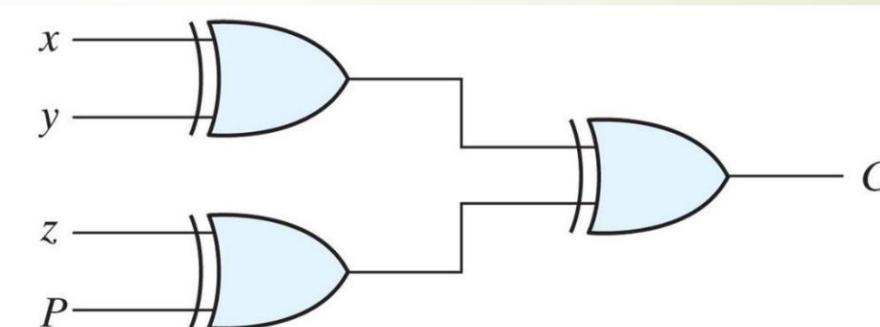
Bilgi **çift parite** ile iletildiğinden alınan dört bitin çift sayıda 1'e sahip olması gereklidir.

Alınan dört bitin tek sayıda 1'i varsa, iletim sırasında bir hata oluşur ; bu, iletim sırasında bir bitin değerinin değiştiğini gösterir.

Parite denetleyicisinin çıktısı, C ile gösterilir , bir hata oluşursa 1'e eşit olacaktır; yani alınan dört bitin tek sayıda 1'i varsa



(a) 3-bit even parity generator



(b) 4-bit even parity checker

Even-Parity-Checker Doğruluk Tablosu

Parite denetleyicisi, özel
VEYA kapılarıyla uygulanabilir

$$C = x \oplus y \oplus z \oplus P$$

Table 3.4
Even-Parity-Checker Truth Table

Four Bits Received				Parity Error Check
x	y	z	P	C
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	1
1	1	1	1	0

Donanım Açıklama Dili (HDL)

Mantık devrelerini tasarlamak için manuel yöntemler uygulanabilir sadece devre küçük olduğunda

Prototip entegre devreler çok pahalıdır ve zaman alır tüketmek inşa etmek

Donanım **tanımlama dili (HDL)**, dijital sistemlerin donanımını metinsel bir biçimde tanımlayan bilgisayar tabanlı bir dildir

Tasarım girişi, donanımda uygulanacak işlevselligin HDL tabanlı bir tanımını oluşturur

105

Donanım Açıklama Dili (HDL)

Mantık simülasyonu, bir dijital sistemin davranışını bir bilgisayar.

Tasarımdaki işlevsel hataları tespit eder

Tasarımın işlevsellliğini test etmeye **test tezgahı** denir

Tüm vakaları denedim mi?

Her yolu denedim mi?

Her seçenek test ediliyor mu?

Simülasyon iki özelliği kontrol eder

İşlevsel doğruluk

mantık doğru mu

Zamanlama

doğruluğu Mantıksal bağlantı zamanlarının doğru olmasıdır

Donanım Açıklama Dili (HDL)

Bir simülatör, HDL açıklamasını yorumlar ve giriş ve çıkış sinyal değerlerinin zaman sıralı dizisi gibi okunabilir çıktı üretir .

Mantık sentezi (derleme), HDL'de tanımlanan bir dijital sistem modelinden fiziksel bileşenlerin ve bunların birbirleriyle olan bağlantılarının (netlist olarak adlandırılır) bir listesini türetme sürecidir .

Bir devrenin elemanlarını ve yapısını açıklayan bir veritabanı oluşturur

Veritabanı, fiziksel bir veri tabanının nasıl oluşturulacağını belirtir HDL'de yapılan ifadelerle tanımlanan işlevselligi silikonda uygulayan entegre devre .

Donanım Açıklama Dili (HDL)

Zamanlama doğrulaması , üretilen entegre devrenin doğruluğunu teyit eder. belirli bir hızda çalışacaktır.

Bir devredeki her mantık kapısının bir [yayılma gecikmesi olduğundan](#), bir devrenin girişindeki bir sinyal geçisi, devrenin çıkışındaki mantık değerinde hemen bir değişikliğe neden olamaz.

Zamanlama doğrulaması, her sinyal yolunun doğru olmadığını doğrulamak için kontrol eder. yayılma gecikmesinden dolayı tehlikeye atılmış

VLSI devre tasarımında, [hata simülasyonu](#), ideal bir devrenin davranışını , işlem kaynaklı bir hata içeren bir devrenin davranışıyla karşılaştırır

Arıza simülasyonu, arızalı devre ile arızasız devre arasındaki farkı ortaya çıkarmak için kullanılabilen giriş uyarılarını tanımlamak için kullanılır

Yalnızca iyi cihazların müşteriye gönderilmesini sağlamak için üretilen cihazları test etmek amacıyla test desenleri kullanılacaktır

HDL Türleri

Desteklenen iki standart HDL vardır
IEEE

VHDL

V, VHSIC (Çok Yüksek Hızlı Entegre) anlamına gelir
(Devre)

VHDL, Verilog'dan öğrenmesi daha zordur.

Verilog

Verilog HDL, başlangıçta 1995'te standart HDL olarak onaylandı (2001, 2005'te revize edildi)

109

HDL Modül Beyanı 100 anahtar kelime

Anahtar kelimeler

önceden tanımlanmış küçük harfli tanımlayıcılardır

Örn: modül, modül sonu, giriş, çıkış, tel, ve, veya, ve değil Yorumlar

// yorumlar için kullanılır /*

*/ çok satırlı yorumlar için Verilog

büyük/küçük harfe duyarlıdır

not, NOT ile aynı şey değildir

Tanımlayıcılar bir alfabetik karakter veya alt çizgi ile başlamalıdır, ancak bir sayı ile başlayamazlar .

Modül terimi , **modül** anahtar sözcük çifti

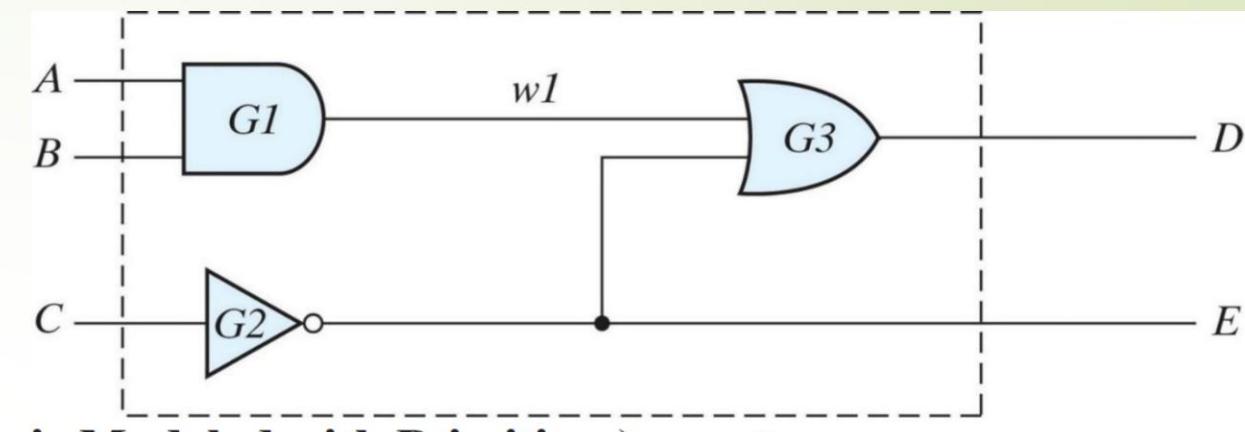
ile çevrelenen metni ifade eder .

...modül **sonu**

Anahtar sözcük **modülü** tarafından bildirilir ve her zaman sonlandırılmalıdır
anahtar kelime **endmodule**

HDL Modül Beyanı

HDL'yi gösteren devre



HDL Example 3.1 (Combinational Logic Modeled with Primitives)

```
// Verilog model of circuit of Figure 3.35. IEEE 1364–1995 Syntax

module Simple_Circuit (A, B, C, D, E);
    output          D, E;
    input           A, B, C;
    wire            w1;
    and             G1 (w1, A, B); // Optional gate instance name
    not             G2 (E, C);
    or              G3 (D, w1, E);
endmodule
```

111

HDL Modül Beyanı

Bir modülün port **listesi** arayüzüdür
modül ve çevresi arasında.

Bu örnekte portlar devrenin giriş ve çıkışlarıdır .

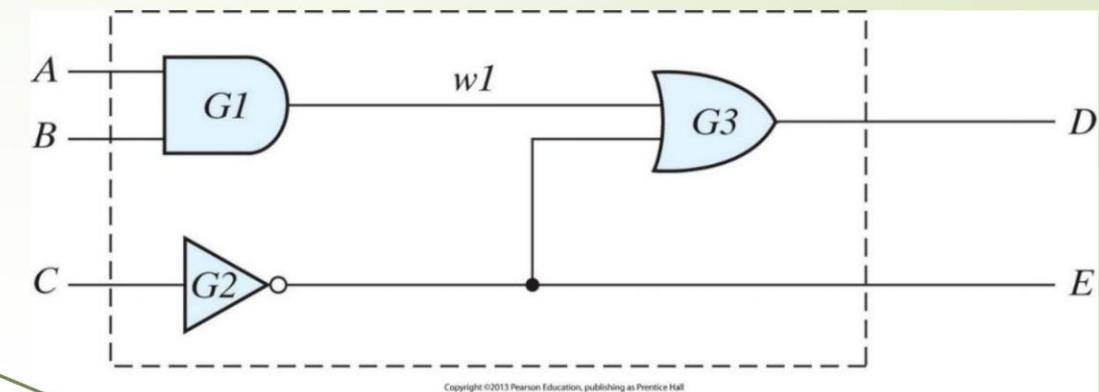
Port listesi parantez içine alınır ve virgüler
öğeleri ayırmak için kullanılır
liste

İfade noktalı virgülle sonlandırılır
();

Anahtar sözcükler **giriş** ve **çıkış**, hangi portların giriş ve
hangilerinin çıkış olduğunu belirtir

Dahili bağlantılar kablo olarak beyan edilir .

Her biri tanımlayıcı bir anahtar sözcüğü (ve, değil,
veya) tanımlanan **ilkel kapılar**



HDL Example 3.1 (Combinational Logic Modeled with Primitives)

// Verilog model of circuit of Figure 3.35. IEEE 1364–1995 Syntax

```
module Simple_Circuit (A, B, C, D, E);
    output D, E;
    input A, B, C;
    wire w1;

    and G1 (w1, A, B); // Optional gate instance name
    not G2 (E, C);
    or G3 (D, w1, E);

endmodule
```

Noktalı virgül yok

Kapılarının her biri kapı örneği olarak adlandırılır

HDL Modül Beyanı

Önceden tanımlanmış ilkel öğeler bildirilmez, çünkü bunlar Tanım dil tarafından belirlenir ve kullanıcı tarafından değiştirilemez.

Örneklemeyi oluşturan ifadelerin **sıralı sıralaması** Modeldeki kapıların **hiçbir önemi yoktur** ve bir hesaplama dizisi belirtmez.

Verilog modeli **tanımlayıcı** bir **modeldir**

Kapı Gecikmeleri

Tüm fiziksel devreler, bir girişin geçiği ile bir çıkışın ortaya çıkan geçiği arasında bir **yayılma gecikmesi** gösterir.

Verilog'da, bir kapının yayılma gecikmesi şu şekilde belirtilir:
zaman birimleri açısından ve **#** simbolü ile

Zaman gecikmeleriyle ilişkili sayılar

Bir zaman biriminin fiziksel zamanla ilişkisi şu şekilde yapılır:
'zaman ölçüği derleyici yönergesi '

Derleyici yönergeleri (**'**) ters tırnak işaretini veya vurgu işaretileyile başlar.

Örn:

'zaman ölçüği 1ns/100ps

İlk sayı zaman gecikmeleri için ölçüm birimini belirtir

İkinci sayı, gecikmelerin yuvarlanacağı **hassasiyeti** belirtir; bu durumda 0,1 ns'ye yuvarlanır.

Kapı Gecikmeleri

ve , veya ve 10 , ve kapılar 30, 20'lik bir zaman gecikmesine sahip değildir , ns

HDL Example 3.2 (Gate-Level Model with Propagation Delays)

```
// Verilog model of simple circuit with propagation delay

module Simple_Circuit_prop_delay (A, B, C, D, E);
    output D, E;
    input A, B, C;
    wire w1;

    and          #(30) G1 (w1, A, B);
    not          #(10) G2 (E, C);
    or           #(20) G3 (D, w1, E);

endmodule
```

Tablo 3.5 Gecikmeden Sonra Kapıların Çıkışı

Devre simüle edilirse ve girişler A'dan değişirse

A , B , C = 1

ve #(30)

#(10) değil

veya #(20)

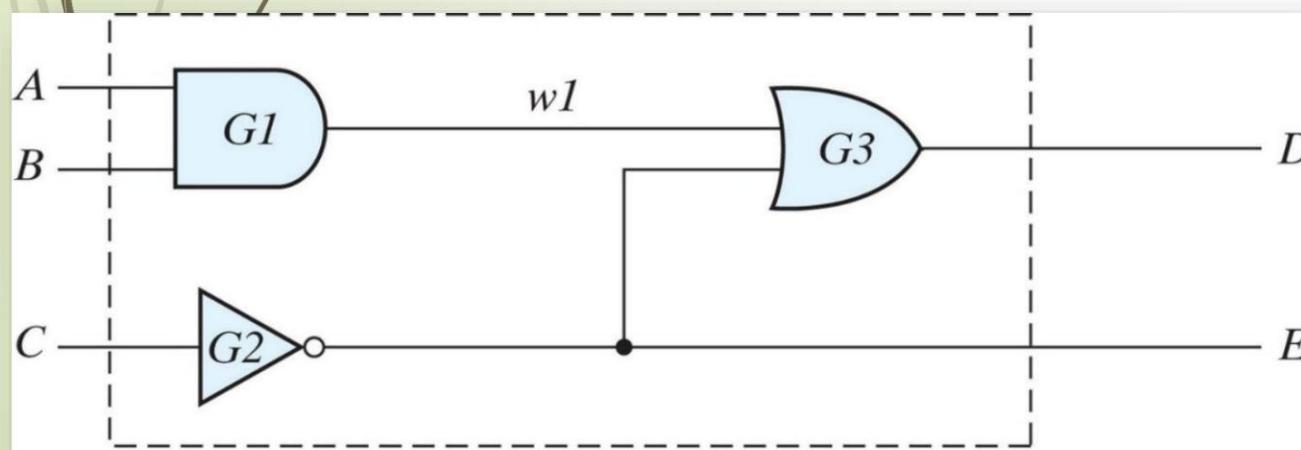


Table 3.5
Output of Gates after Delay

Time Units (ns)	Input			Output		
	A	B	C	E	w1	D
Initial	—	—	—	0	0	0
Change	—	—	—	1	1	1
10	—	—	—	1	1	1
20	—	—	—	0	0	1
30	—	—	—	0	0	1
40	—	—	—	0	1	0
50	—	—	—	0	1	0

116

Test tezgahı

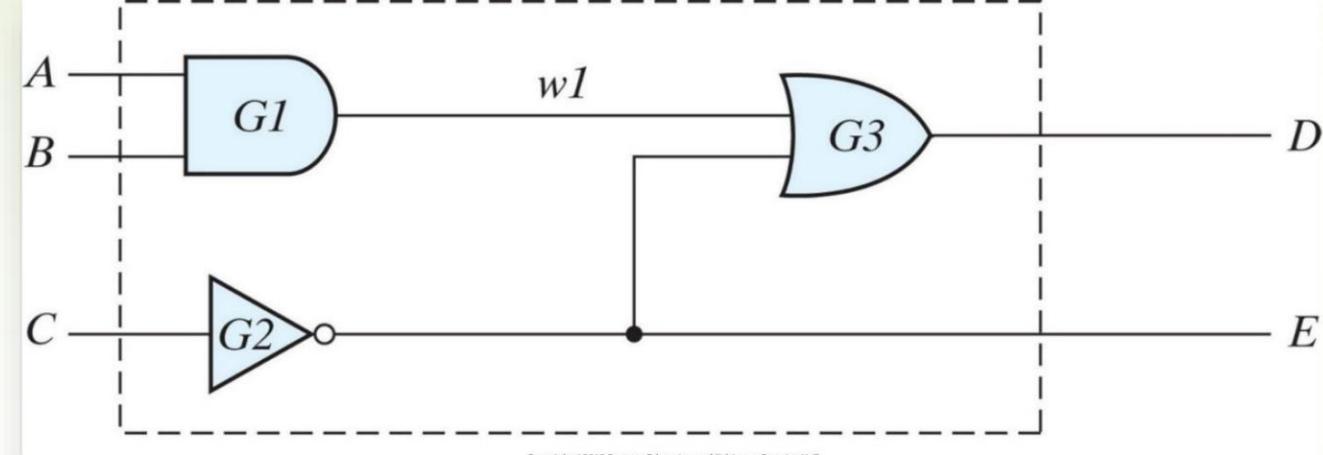
Test tezgahlarının giriş veya çıkış portu yoktur

Çevre ile etkileşim yok

Devreye girişler reg anahtar
sözcüğüyle bildirilir
çıktılar **wire anahtar** sözcüğüyle
bildirilir

Modül

Simple_Circuit_prop_delay , benzersiz
örnek adı **M1** ile örneklenir



```
// Test bench for Simple_Circuit_prop_delay
module t_Simple_Circuit_prop_delay;
  wire D, E;
  reg A, B, C;

  Simple_Circuit_prop_delay M1 (A, B, C, D, E); // Instance name required

  initial
    begin
      A = 1'b0; B = 1'b0; C = 1'b0;
      #100 A = 1'b1; B = 1'b1; C = 1'b1;
    end

  initial #200 $finish;
endmodule
```

117

Test tezgahı

Başlangıç anahtar sözcüğü , simülasyon başlatıldığında yürütülmeye başlayan bir dizi ifadeyle kullanılır

İfadeler şu şekilde yürütülür:

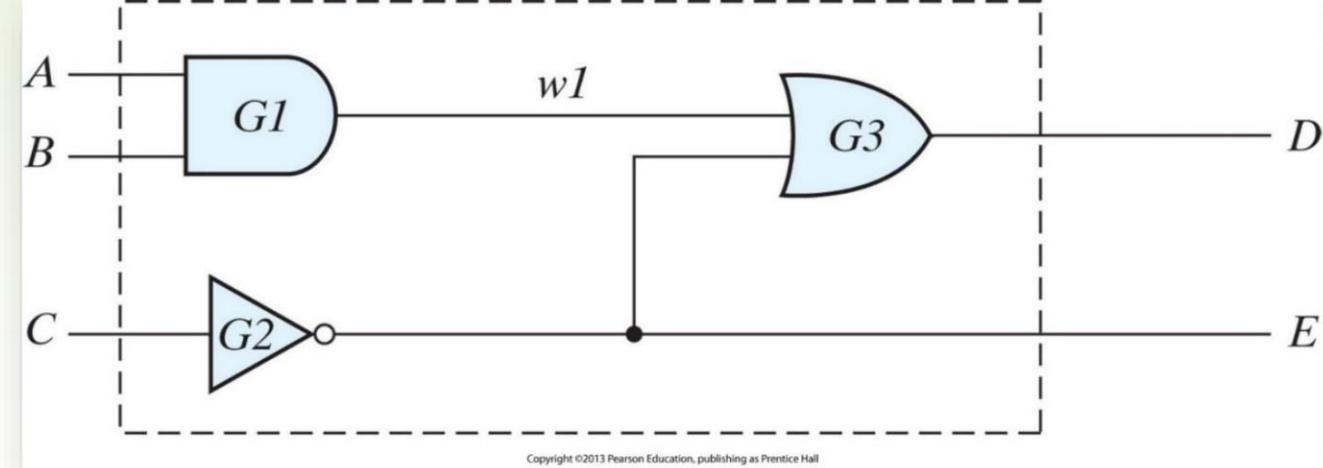
sıra, soldan sağa, yukarıdan aşağıya

`1'b0` , 0 değerine sahip bir ikili rakamı ifade eder

100 ns sonra girişler A, B, C = 1 olarak değişir.

100 ns daha sonra,

simülasyon 200 ns'de sonlanır



```
// Test bench for Simple_Circuit_prop_delay
module t_Simple_Circuit_prop_delay;
wire D, E;
reg A, B, C;

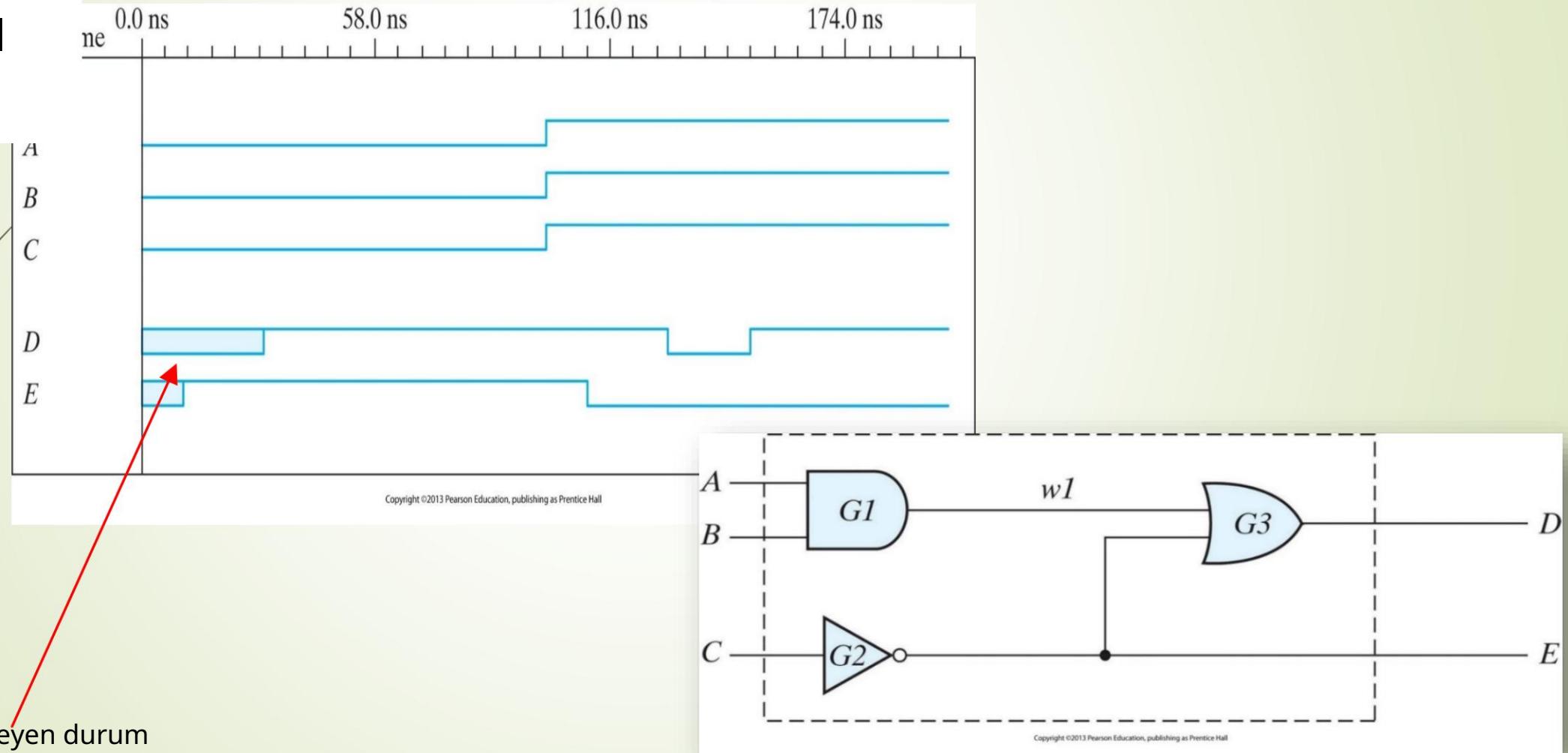
Simple_Circuit_prop_delay M1 (A, B, C, D, E); // Instance name required

initial
begin
    A = 1'b0; B = 1'b0; C = 1'b0;
    #100 A = 1'b1; B = 1'b1; C = 1'b1;
end

initial #200 $finish;
endmodule
```

ŞEKİL 3.36 HDL Örnek 3.3'ün simülasyon çıktıları

ve #(30)
#(10) değil
veya #(20)



Boolean İfadeleri

$$E = A + BC + B'D$$
$$F = B'C + BC'D'$$

Boolean İfadesi tarafından takip edilen anahtar sözcük **ataması**

Verilog, AND, OR ve NOT (tamamlayıcı) için **(&&)**, **(|)** ve **(!)** sembollerini kullanır

HDL Example 3.4 (Combinational Logic Modeled with Boolean Equations)

```
// Verilog model: Circuit with Boolean expressions

module Circuit_Boolean_CA (E, F, A, B, C, D);
    output      E, F;
    input       A, B, C, D;

    assign E = A || (B && C) || ((!B) && D);
    assign F = ((!B) && C) || (B && (!C) && (!D));
endmodule
```

ilkel anahtar
sözcüğü
ile
tanımlanır ,
ardından bir ad ve
port listesi
gelir

Sadece bir çıktı olabilir
ve bu çıktı port listesinde
ilk sırada listelenmeli ve
output anahtar
sözcüğüyle
bildirilmelidir

Kullanıcı Tarafından Tanımlanan İlkel Öğeler

HDL Example 3.5 (User-Defined Primitive)

// Verilog model: User-defined Primitive

```
primitive UDP_02467 (D, A, B, C);
    output D;
    input A, B, C;
```

//Truth table for D 5 f (A, B, C) 5 Σ(0, 2, 4, 6, 7);

table

			:	D	// Column header comment
0	0	0	:	1;	Giriş değerleri, iki nokta üst üste (:) ile biten sırayla listelenir.
0	0	1	:	0;	
0	1	0	:	1;	
0	1	1	:	0;	
1	0	0	:	1;	
1	0	1	:	0;	
1	1	0	:	1;	
1	1	1	:	1;	

endtable

endprimitive

Gerçeklik
tablosu
anahtar
kelimeler
tablosu ve
son tablonun
icin yerlestirilmiştır

Herhangi bir
sayda girdi olabilir.
Girişte
listelendikleri sıra

beyanlar tabloda
verilen değerlere
uygun olmalıdır

Cıktı her zaman
satırda son girdidir
ve noktalı virgül (:)
ile takip edilir

UDP bildirimi
ndprimitive
anahtar sözcüğüyle
son bulur .

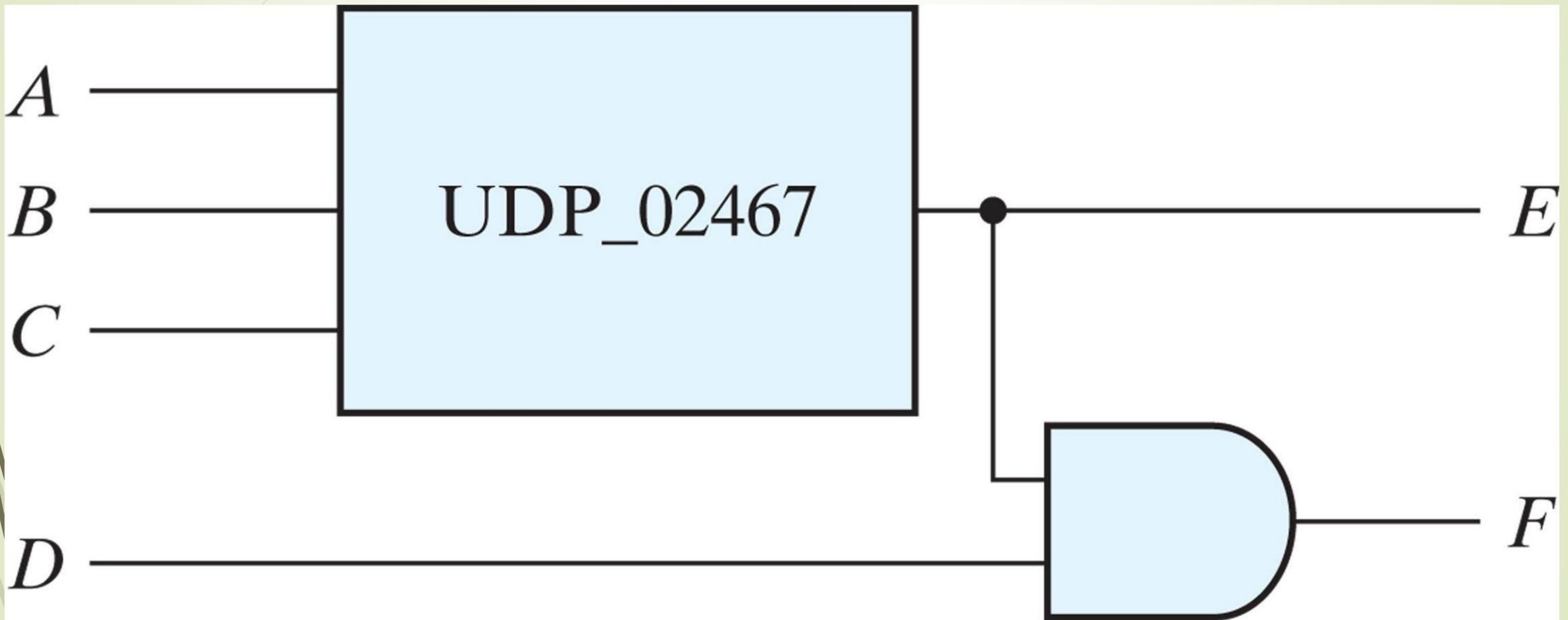
```
// Instantiate primitive
```

```
// Verilog model: Circuit instantiation of Circuit_UDP_02467
```

```
module Circuit_with_UDP_02467 (e, f, a, b, c, d);
    output      e, f;
    input       a, b, c, d

    UDP_02467      (e, a, b, c);
    and           (f, e, d);          // Option gate instance name omitted
endmodule
```

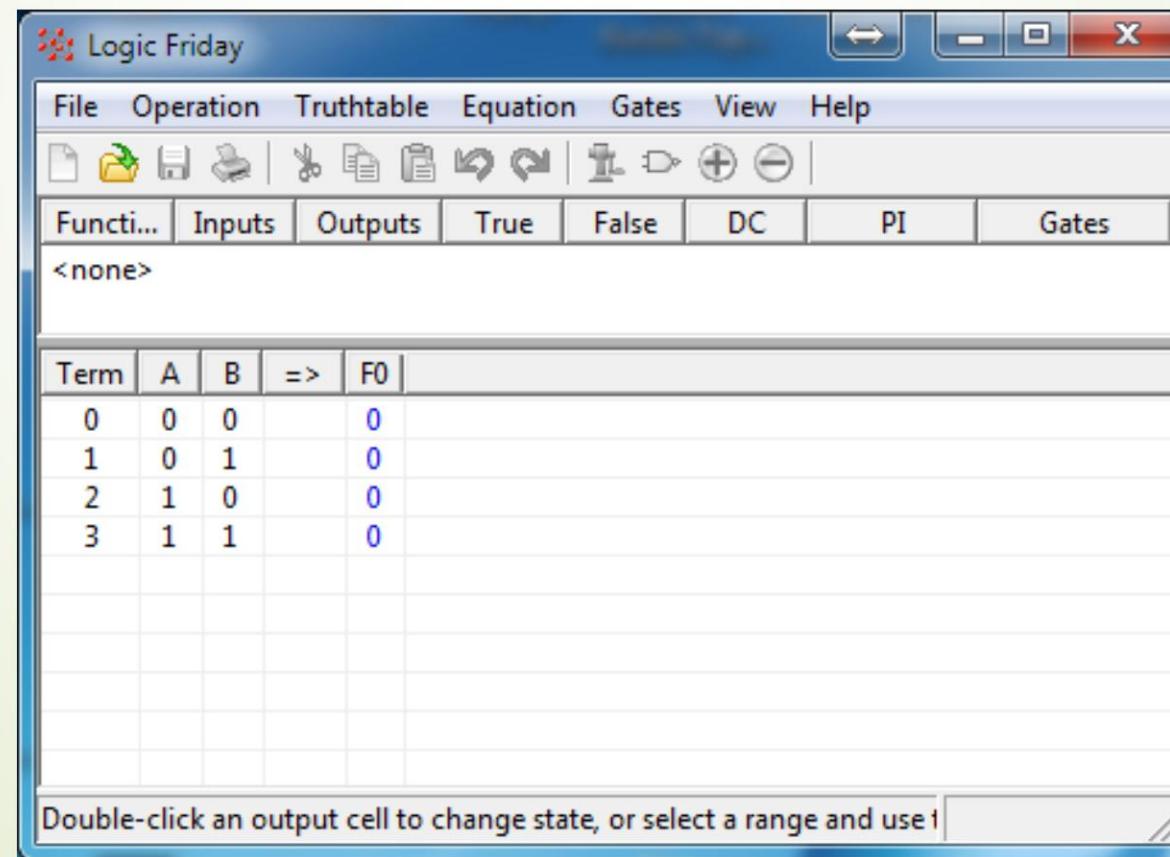
Kullanıcı Tarafından Tanımlanan İlkel Öğeler



123

Mantık Cuma

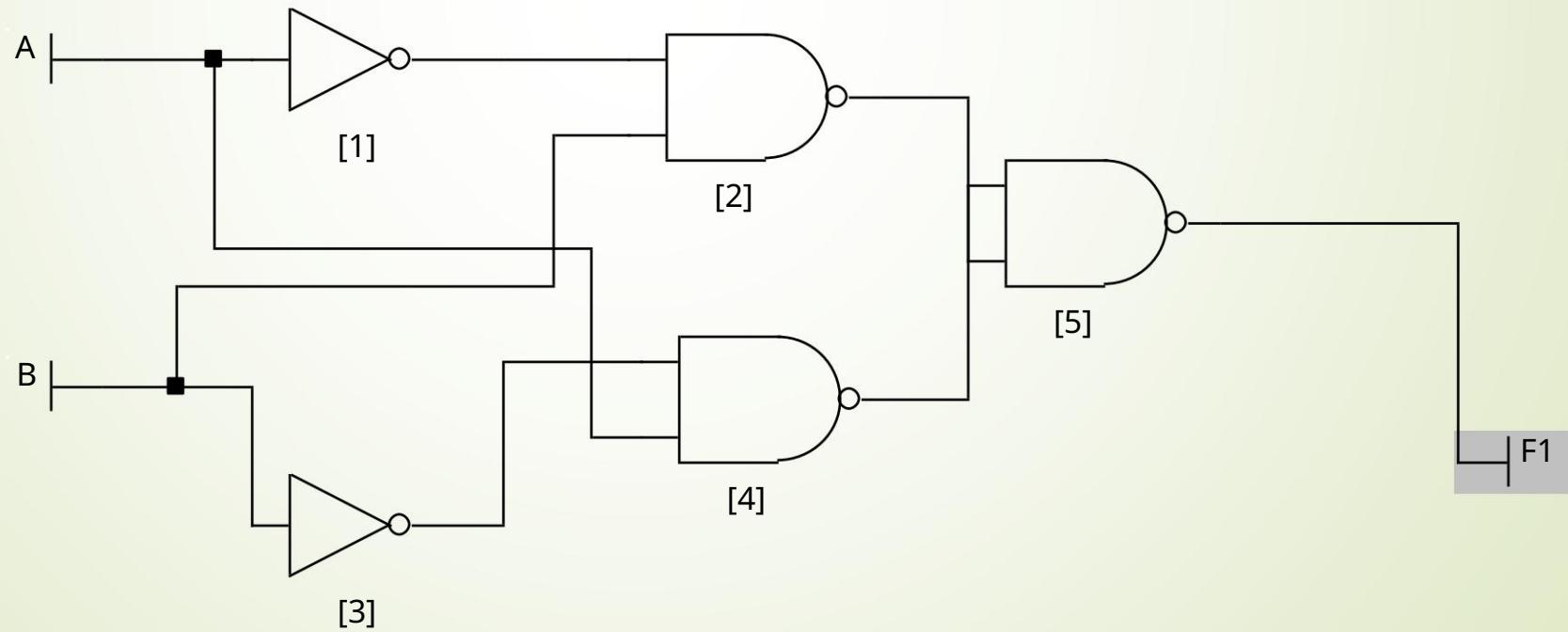
Logic Friday, standart IC paketlerine dayalı eski dijital mantık devreleriyle çalışan öğrenciler, amatörler ve mühendisler için ücretsiz bir araçtır.



124

Mantık Cuma (XOR)

XOR Örneği



125

İkarus Verilog

Icarus Verilog, bir Verilog simülasyon ve sentez aracıdır.
1995, 2001 ve 2005 sürümlerini destekler.
standart

Derleyici olarak çalışır

İkarus Verilog

Derleyici, şu şekilde adlandırılan bir ara form üretebilir:

vvv meclisi.

vvp derlenmiş “vvp assembly”yi yürütür

VCD biçimli günlük dosyasını çıktı olarak

yazar gtkwave, VCD (ve diğer) dosyalarını grafiksel olarak
görüntüleyen açık kaynaklı bir dalga formu görüntüleyicisidir

<http://gtkwave.sourceforge.net/>

Verilog Hata Ayıklama

Sabit sözdizimi:

1'b0 = 1 bit uzunlığında, ikili 0

8'b01011010 = 8'h5A = 8'd90

\$display – C'deki printf gibi

`$display($time, " Sayım %d olarak değiştirildi", sayı);`

\$dumpfile ve \$dumpvars

tüm sinyal değerlerini daha sonra görüntülemek üzere bir dosyaya aktarın

`$dumpfile("çıkıtı.log")`

`$dumpvars(0, en üst_seviye_modülü)`

128

Referanslar

Linda Null, Julia Lobur, Bilgisayar Organizasyonunun Temelleri ve
Mimarlık

Mano'nun ders kitabı Slaytlar